Практическая работа №3

Тема: Логистическая регрессия

**Задание:**

1 Изучить теоретический материал

2 Выполнить задачу построения логистической регрессии в Python

**1.Теоретический материал**

Что такое логистическая регрессия?

Логистическая регрессияэто статистический метод для анализа набора данных, в котором есть одна или несколько независимых переменных, которые определяют результат. Результат измеряется с помощью дихотомической переменной (в которой есть только два возможных результата). Он используется для прогнозирования двоичного результата (1/0, Да / Нет, Истина / Ложь) с учетом набора независимых переменных.

Вы также можете рассматривать логистическую регрессию как особый случай линейной регрессии, когда исходная переменная является категориальной, где мы используем логарифм шансов в качестве зависимой переменной. Проще говоря, он предсказывает вероятность возникновения события путем подгонки данных клогитфункция.

Помните, что в некоторых случаях зависимые переменные могут иметь более двух результатов, например, в браке / не замужем / в разводе, такие сценарии классифицируются какполиномиальная логистическая регрессия.Хотя они работают одинаково, чтобы предсказать результат.

Несколько знакомых примеров логистической регрессии:

Некоторые выдающиеся примеры, такие как:

Фильтр спама в электронной почте: спам / нет спама

Обнаружение мошенничества: транзакция является мошеннической, да / нет

Опухоль: доброкачественная / злокачественная

Как работает логистическая регрессия?

Поскольку нам ясно, что логистическая регрессия в основном делает прогнозы для решения проблем, которые требуют оценки вероятности в виде результата, в форме 0/1. Логистическая регрессия - чрезвычайно эффективный механизм для вычисления вероятностей. Поэтому вам должно быть любопытно понять, как всегда получается значение 0 или 1. Чтобы понять больше, давайте попробуем расшифровать некоторые математические основы логистической регрессии.

Логистическая модель: сигмовидная функция

Давайте попробуем понять логистическую регрессию, понимая логистическую модель. Как и в случае линейной регрессии, давайте представим нашу гипотезу (Предсказание зависимой переменной) в классификации. В классификации наше представление гипотезы, которое пытается предсказать двоичный результат или o или 1, будет выглядеть так:

hθ (x) = g (θ T x) = 1/1 + e − θ T x,

Здесь g (z) = 1 / (1 + e ^ −z) называется lОгистическая функция или сигмовидная функция:

(г): представление логистической функции, которую мы также называем сигмовидной функцией. Из приведенного выше визуального представления сигмовидной функции мы можем легко понять, как эта кривая описывает многие реальные ситуации, такие как рост населения. На начальных этапах это показывает экспоненциальный рост, но через некоторое время, из-за конкуренции за определенные ресурсы (горлышко бутылки), скорость роста снижается, пока не достигнет тупиковой ситуации, и рост не будет

Вопрос здесь в том, как этологит(сигмоидальная функция) помогает нам определить вероятность классификации данных по различным классам. Давайте попробуем понять, как рассчитывается наша функция logit, что даст нам некоторую ясность

Математика за логистической функцией:

Шаг 1: Классификация входных данных должна быть в классе ноль или единица.

Во-первых, нам нужно вычислить вероятность того, что наблюдение принадлежит классу 1 (мы также можем назвать его положительным классом), используя функцию логистического отклика. В этом случае наш параметр z, как видно из приведенной ниже функции logit.

ЦСИ

Коэффициенты бета 0, бета 1, бета K, как показано на рис. Выше, выбраны, чтобы максимизировать вероятность прогнозирования высокой вероятности для наблюдений, принадлежащих к классу 1,

И прогнозирование малой вероятности для наблюдений, фактически принадлежащих к классу 0. Выше рис показывает, что мы пытаемся сделать с коэффициентами.

Log Odds (функция Logit):

Вышеприведенное объяснение также может быть понято с точки зрения логарифмических коэффициентов, что является своего рода пониманием вероятности классификации элементов на классы (1 или 0) с помощьюСТАВКИ:

ЦСИ

Эти шансы, которые напоминают сходство с линейной регрессией, называютсялогит.

Таким образом, логит - это журнал шансов, а шансы - это функция P. В логистической регрессии мы находим

logit (P) = a + bX,

Шаг 2: Определение граничных значений для шансов

Теперь мы определим границу порога, чтобы четко классифицировать каждое заданное входное значение в один из классов.

Мы можем выбрать пороговое значение в соответствии с бизнес-проблемой, которую мы пытаемся решить, как правило, которая находится в районе 0,5 Таким образом, если ваши значения вероятности окажутся> 0,5, мы можем классифицировать такое наблюдение в тип класса 1, а остальные в класс 0.Выбор порогового значения обычно основывается на типах ошибок, которые бывают двух типов:ложные срабатывания и ложные отрицания.

Ложно-положительная ошибка возникает, когда модель прогнозирует класс 1, но наблюдение фактически принадлежит классу 0. Ложно-отрицательная ошибка допускается, когда модель прогнозирует класс 0, но наблюдение фактически принадлежит классу 1. Идеальная модель будет классифицировать все правильно классифицирует: все 1 (или истины) как 1, и все 0 (или ложь) как 0. Таким образом, мы имели бы FN = FP = 0.

Влияние пороговых значений:

1.Более высокое пороговое значение

Предположим, если P (y = 1)> 0,7. Модель является более строгой при классификации как 1, и, следовательно, будет сделано больше ошибок ложного отрицания.

2. Нижнее пороговое значение:

Предположим, если P (y = 1)> 0,3.

Модель теперь менее строгая, и мы классифицируем больше примеров как класс 1, поэтому мы делаем больше ошибок ложных срабатываний.

Путаница Матрица: путь к Choose Эффективное пороговое значение:

Матрица путаницы, также известная как матрица ошибок, является предиктором производительности модели для задачи классификации. Количество правильных и неправильных прогнозов суммируется со значениями количества и разбивается по каждому классу. Это лежит в основе путаницы.

Матрица путаницы показывает, каким образом ваша модель классификации находится в замешательстве, когда она делает прогнозы для наблюдений, она помогает нам измерить тип ошибки, которую делает наша модель, при классификации наблюдения по различным классам.

Матрица путаницы

Ключевые части матрицы путаницы:

Истинный Позитив(TP): Это относится к случаям, в которых мы предсказывали «ДАИ наш прогноз был на самом делеПРАВДА

(Например, у пациента на самом деле диабет, и вы предсказали, что это правда)

Правда отрицательный(TN): это относится к случаям, в которых мы предсказывали «НЕТИ наш прогноз был на самом делеПРАВДА

(Например, у пациента нет диабета, и вы предсказали то же самое.)

Ложно положительный(FP): это относится к случаям, в которых мы предсказывали «ДА», Но наш прогноз оказалсяЛОЖНЫЙ

(У пациента не было диабета, но наша модель предсказала, что он / она диабетик)

Ложный Отрицательный(FN): это относится к случаям, в которых мы предсказывали «НЕТНо наш прогноз оказалсяЛОЖНЫЙ

Ключевые показатели обучения из матрицы путаницы:

Матрица путаницы помогает нам изучать следующие метрики, помогая нам измерять производительность логистической модели.

Точность:

Это отвечает:

В целом, как часто верен классификатор?

Точность = (TP + TN) / общее количество засекреченных предметов = (TP + TN) / (TP + TN + FP + FN)

Точность:

Это отвечает:

Когда это предсказывает да, как часто это правильно?

Точность обычно используется, когда целью являетсяограничить количество ложных срабатываний(ФП). Например, с помощью алгоритма фильтрации спама, где наша цель состоит в том, чтобы свести к минимуму количество реальных электронных писем, которые классифицируются как спам

Точность = TP / (TP + FP)

Отзыв:

Это отвечает:

Когда это на самом деле положительный результат, как часто он предсказывает правильно?

Рассчитывается как,

Напомним = TP / (TP + FN), также известный как чувствительность.

f1-счет:

Это просто гармоническое среднее точности и напоминания:

Рассчитывается как,

f1-оценка = 2 \* ((точность \* отзыв) / (точность + отзыв))

Поэтому, когда вам нужно принять во внимание как точность, так и вспомнить, этот показатель f1 является полезным показателем для измерения. Если вы попытаетесь оптимизировать только отзыв, ваш алгоритм будет предсказывать, что большинство примеров будет принадлежать положительному классу, но это приведет к множеству ложных срабатываний и, следовательно, к низкой точности. Кроме того, если вы попытаетесь оптимизировать точность, ваша модель будет предсказывать очень мало примеров как положительные результаты (те, которые имеют наибольшую вероятность), но отзыв будет очень низким. Так что может быть полезно сбалансировать и рассмотреть оба варианта и увидеть результат.

Площадь AUC под кривой:

Еще одна полезная метрика для оценки и сравнения прогностических моделей - это кривая ROC.

В статистикеРабочая характеристика приемника (ROC)или кривая ROC - это графический график, который иллюстрирует производительность системы двоичного классификатора при изменении порога дискриминации. Кривая создается путем нанесения истинного положительного показателя (чувствительности) на уровень ложного положительного результата (1 - специфичность) при различных настройках пороговых значений.

Кривые ROC - хороший способ увидеть, как любая прогностическая модель может отличить истинные положительные стороны от отрицательных. Кривая ROC отображает чувствительность и специфичность для каждого возможного предела правила принятия решения между 0 и 1 для модели.

Где,

Специфичность или истинный отрицательный показатель= TN / (TN + FP)

Чувствительность или истинно положительный показатель= TP / (TP + FN)

ТакFPR, ложноположительный показатель = 1 – специфичность

Интуиция за кривой ROC:

Эта модель, которая предсказывает случайно, будет иметь ROC-кривую, которая выглядит как диагональная зеленая линия (как показано выше на рисунке). Это не дискриминационная модель. Чем дальше кривая от диагональной линии, тем лучше модель различает положительные и отрицательные значения в целом.

Типы логистической регрессии:

Бинарная логистическая регрессия:У этого есть только два возможных результата. Пример - да или нет

Полиномиальная логистическая регрессияУ этого есть три или больше номинальных категории. Пример - кошка, собака, слон.

Порядковый логистический регресс- У него есть три или более порядковых категорий, порядковое значение которых означает, что категории будут в порядке. Пример - пользовательские рейтинги (1–5).

Поскольку мы поняли некоторые важные оговорки, связанные с логистической регрессией, пришло время взять некоторое практическое понимание на простом примере:

Реализация логистической регрессии:

Мы собираемся охватить это упражнение по созданию модели в следующие шаги:

Чтение данных

Анализ данных (базовый EDA / описательный анализ)

Обучить и проверить (разбить пример данных на два набора)

Отчет о точности (измерение производительности модели с использованием матрицы путаницы, о которой мы говорили выше)

**2. Задача построения логистической регрессии в Python**

**Пошаговое построение логистической** регрессии в Python

**Условия логистической регрессии**

* Бинарная логистическая регрессия требует, чтобы зависимая переменная также была бинарной.
* Для бинарной регрессии фактор уровня 1 зависимой переменной должен представлять желаемый вывод.
* Использоваться должны только значимые переменные.
* Независимые переменные должны быть независимы друг от друга. Это значит, что модель должна иметь малую мультиколлинеарность или не иметь её вовсе.
* Независимые переменные связаны линейно с логарифмическими коэффициентами.
* Логистическая регрессия требует больших размеров выборки.

Держа в уме все перечисленные условия, давайте взглянем на наш набор данных.

**Данные**

Набор данных взят с [репозитория машинного обучения UCI](http://archive.ics.uci.edu/ml/index.php%22%20%5Ct%20%22_blank) и относится к прямым маркетинговым кампаниям (телефонный обзвон) португальского банковского учреждения. Цель классификации в прогнозировании успеха подписки клиента (1/0) на срочный депозит (переменная y). Загрузить этот набор данных можно [здесь](https://raw.githubusercontent.com/madmashup/targeted-marketing-predictive-engine/master/banking.csv).

import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear\_model import LogisticRegression
from sklearn.model\_selection import train\_test\_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color\_codes=True)

Эти данные предоставляют информацию о клиентах банка, которая включает 41,188 записей и 21 поле.



Рис. 1

**Вводные переменные**

1. age — возраст (число);
2. job — вид должности (категории: “admin” (администратор), “blue-collar” (рабочий), “entrepreneur” (мелкий предприниматель), “housemaid” (горничная), “management” (руководитель), “retired” (на пенсии), “self-employed” (самозанятый), “services” (сфера услуг), “student” (учащийся), “technician” (техник), “unemployed” (не трудоустроен), “unknown” (неизвестно));
3. marital — семейное положение (категории: “divorced” (разведён), “married” (замужем/женат), “single” (холост/не замужем), “unknown” (неизвестно));
4. education — образование (категории: “basic.4y”, “basic.6y”, “basic.9y” (базовое), “high.school” (высшая школа), “illiterate” (без образования), “professional.course” (профессиональные курсы), “university.degree” (университетская степень), “unknown” (неизвестно));
5. default — имеет ли просроченные кредиты (категории: “no” (нет), “yes” (да), “unknown” (неизвестно));
6. housing — имеет ли жилищный кредит (категории: “no” (нет), “yes” (да), “unknown” (неизвестно));
7. loan — имеет ли личный кредит (категории: “no” (нет), “yes” (да), “unknown” (неизвестно));
8. contact — вид связи (категории: “cellular” (мобильный), “telephone” (стационарный));
9. month — месяц последнего обращения (категории: “jan” (январь), “feb” (февраль), “mar” (март), …, “nov” (ноябрь), “dec” (декабрь));
10. day\_of\_week — день недели последнего обращения (категории: “mon” (понедельник), “tue” (вторник), “wed” (среда), “thu” (четверг), “fri” (пятница));
11. duration — продолжительность последнего обращения в секундах (число). Важно: этот атрибут оказывает сильное влияние на вывод (например, если продолжительность=0, тогда y=’no’). Продолжительность не известна до момента совершения звонка, а по его завершению y будет, очевидно, известна. Следовательно этот вводный параметр должен включаться только в целях эталонного тестирования, для получения же реалистичной модели прогноза его следует исключать.
12. campaign — число обращений, установленных в процессе этой кампании и для этого клиента (представлено числом и включает последнее обращение);
13. pdays — число дней, прошедших с момента последнего обращения к клиенту во время предыдущей кампании (число; 999 означает, что ранее обращений не было);
14. previous — число обращений, совершённых до этой кампании (число);
15. poutcome — итоги предыдущей маркетинговой кампании (категории: “failure” (провал), “nonexistent” (несуществующий), “success” (успех));
16. emp.var.rate — коэффициент изменения занятости (число);
17. cons.price.idx — индекс потребительских цен (число);
18. cons.conf.idx — индекс потребительского доверия (число);
19. euribor3m — 3-х месячная европейская межбанковская ставка (число);
20. nr.employed — количество сотрудников (число).

**Прогнозируемая переменная (желаемая цель):**

y —подписался ли клиент на срочный вклад (двоично: “1” означает “Да”, “0” означает “Нет”).

Колонка образования в наборе данных имеет очень много категорий, и нам нужно сократить их для оптимизации моделирования. В этой колонке представлены следующие категории:



Рис. 2

Давайте сгруппируем “basic.4y”, “basic.9y” и “basic.6y” и назовём их “basic” (базовое).

data['education']=np.where(data['education'] =='basic.9y', 'Basic', data['education'])
data['education']=np.where(data['education'] =='basic.6y', 'Basic', data['education'])
data['education']=np.where(data['education'] =='basic.4y', 'Basic', data['education'])

После группировки мы получим следующие колонки:



Рис. 3

**Изучение данных**



Рис. 4

count\_no\_sub = len(data[data['y']==0])
count\_sub = len(data[data['y']==1])
pct\_of\_no\_sub = count\_no\_sub/(count\_no\_sub+count\_sub)
print("percentage of no subscription is", pct\_of\_no\_sub\*100)
pct\_of\_sub = count\_sub/(count\_no\_sub+count\_sub)
print("percentage of subscription", pct\_of\_sub\*100)

* ***Процент не подписавшихся — 88,73458288821988;***
* ***Процент подписавшихся — 11,265417111780131.***

Наши классы не сбалансированы, и соотношение не подписавшихся к подписавшимся составляет 89:11. Прежде чем мы перейдём к балансировке, давайте проведём дополнительное исследование.



Рис. 5

***Наблюдения***:

* Средний возраст потребителей, купивших срочный депозит, выше, чем тех, кто его не купил.
* pdays (дни, прошедшие с момента последнего обращения к клиенту) по понятным причинам ниже для тех, кто купил депозит. Чем меньше показатель pdays, тем лучше память о последнем звонке и выше шансы успешной продажи.
* На удивление, показатель campaigns (число обращений или звонков, сделанных во время текущей кампании) меньше для потребителей, купивших срочный депозит.

Мы можем вычислить средние категориальные значения для других категориальных переменных, вроде образования и семейного положения, чтобы получить более точное представление о данных.



Рис. 6



Рис. 7

**Визуализации**

%matplotlib inline
pd.crosstab(data.job,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Job Title')
plt.xlabel('Job')
plt.ylabel('Frequency of Purchase')
plt.savefig('purchase\_fre\_job')



Рис. 8

Частота покупки депозита во многом зависит от должности клиента. Следовательно этот показатель может послужить хорошим фактором прогноза итоговой переменной.

table=pd.crosstab(data.marital,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')
plt.savefig('mariral\_vs\_pur\_stack')



Рис. 9

Семейное же положение не является существенным прогнозирующим фактором.

table=pd.crosstab(data.education,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Education vs Purchase')
plt.xlabel('Education')
plt.ylabel('Proportion of Customers')
plt.savefig('edu\_vs\_pur\_stack')



Рис. 10

Образование выглядит хорошим фактором для прогноза итоговой переменной.

pd.crosstab(data.day\_of\_week,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Frequency of Purchase')
plt.savefig('pur\_dayofweek\_bar')



Рис. 11

День недели может не являться хорошим прогнозирующим фактором.

pd.crosstab(data.month,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Month')
plt.xlabel('Month')
plt.ylabel('Frequency of Purchase')
plt.savefig('pur\_fre\_month\_bar')



Рис. 12

Месяц может оказаться хорошим прогнозирующим фактором.

data.age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('hist\_age')



Рис. 13

Большинство клиентов банка в этом наборе данных находятся в возрасте от 30 до 40 лет.

pd.crosstab(data.poutcome,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Poutcome')
plt.xlabel('Poutcome')
plt.ylabel('Frequency of Purchase')
plt.savefig('pur\_fre\_pout\_bar')



Рис. 14

Poutcome (итоги предыдущей маркетинговой кампании) кажется хорошим прогнозирующим фактором.

**Создание индикаторных переменных**

Это переменные со всего двумя значениями — ноль и единица.

cat\_vars=['job','marital','education','default','housing','loan','contact','month','day\_of\_week','poutcome']
for var in cat\_vars:
 cat\_list='var'+'\_'+var
 cat\_list = pd.get\_dummies(data[var], prefix=var)
 data1=data.join(cat\_list)
 data=data1cat\_vars=['job','marital','education','default','housing','loan','contact','month','day\_of\_week','poutcome']
data\_vars=data.columns.values.tolist()
to\_keep=[i for i in data\_vars if i not in cat\_vars]

Наши итоговые колонки данных будут:

data\_final=data[to\_keep]
data\_final.columns.values



Рис. 15

**Over-Sampling при помощи SMOTE**

После создания наших обучающих данных я увеличу частоту выборки не подписавшихся, используя [алгоритм SMOTE](https://arxiv.org/pdf/1106.1813.pdf) (синтетическая техника дублирования примеров миноритарного класса). На высоком уровне SMOTE:

1. создаёт синтетические образцы на основе выборок минорного класса (не подписавшихся) вместо создания их копий;
2. случайно выбирает одного из ближайших k-соседей и использует его для создания схожих, но случайно изменённых новых сведений.

Мы будем реализовывать [SMOTE в Python](http://imbalanced-learn.org/en/stable/over_sampling.html#smote-variants).

X = data\_final.loc[:, data\_final.columns != 'y']
y = data\_final.loc[:, data\_final.columns == 'y']from imblearn.over\_sampling import SMOTEos = SMOTE(random\_state=0)
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)
columns = X\_train.columnsos\_data\_X,os\_data\_y=os.fit\_sample(X\_train, y\_train)
os\_data\_X = pd.DataFrame(data=os\_data\_X,columns=columns )
os\_data\_y= pd.DataFrame(data=os\_data\_y,columns=['y'])
# Мы можем проверить числа наших данных
print("length of oversampled data is ",len(os\_data\_X))
print("Number of no subscription in oversampled data",len(os\_data\_y[os\_data\_y['y']==0]))
print("Number of subscription",len(os\_data\_y[os\_data\_y['y']==1]))
print("Proportion of no subscription data in oversampled data is ",len(os\_data\_y[os\_data\_y['y']==0])/len(os\_data\_X))
print("Proportion of subscription data in oversampled data is ",len(os\_data\_y[os\_data\_y['y']==1])/len(os\_data\_X))



Рис. 16

Теперь у нас есть идеально сбалансированные данные. Вы могли заметить, что я выполнил over-sampling только для обучающих данных, поскольку в таком случае информация из тестовых данных не используется для создания синтетических сведений, и, следовательно, не произойдёт её утечки в обучающую модель.

**Рекурсивное устранение признаков**

[Рекурсивное устранение признаков](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html)(RFE) основывается на повторяющемся конструировании модели и выборе лучше всех или хуже всех выполняемого признака, отделения этого признака и повторения цикла с оставшимися. Этот процесс применяется, пока в наборе данных не закончатся признаки. Цель RFE заключается в отборе признаков посредством рекурсивного рассмотрения всё меньшего и меньшего их набора.

data\_final\_vars=data\_final.columns.values.tolist()
y=['y']
X=[i for i in data\_final\_vars if i not in y]from sklearn.feature\_selection import RFE
from sklearn.linear\_model import LogisticRegressionlogreg = LogisticRegression()rfe = RFE(logreg, 20)
rfe = rfe.fit(os\_data\_X, os\_data\_y.values.ravel())
print(rfe.support\_)
print(rfe.ranking\_)



Рис. 16

При помощи RFE мы выбрали следующие признаки: “euribor3m”, “job\_blue-collar”, “job\_housemaid”, “marital\_unknown”, “education\_illiterate”, “default\_no”, “default\_unknown”, “contact\_cellular”, “contact\_telephone”, “month\_apr”, “month\_aug”, “month\_dec”, “month\_jul”, “month\_jun”, “month\_mar”, “month\_may”, “month\_nov”, “month\_oct”, “poutcome\_failure”, “poutcome\_success”.

cols=['euribor3m', 'job\_blue-collar', 'job\_housemaid', 'marital\_unknown', 'education\_illiterate', 'default\_no', 'default\_unknown',
 'contact\_cellular', 'contact\_telephone', 'month\_apr', 'month\_aug', 'month\_dec', 'month\_jul', 'month\_jun', 'month\_mar',
 'month\_may', 'month\_nov', 'month\_oct', "poutcome\_failure", "poutcome\_success"]
X=os\_data\_X[cols]
y=os\_data\_y['y']

**Реализация модели**

import statsmodels.api as sm
logit\_model=sm.Logit(y,X)
result=logit\_model.fit()
print(result.summary2())



Рис. 17

p-значения для большинства переменных меньше 0.05, за исключением всего четырёх, следовательно мы их удалим.

cols=['euribor3m', 'job\_blue-collar', 'job\_housemaid', 'marital\_unknown', 'education\_illiterate',
 'month\_apr', 'month\_aug', 'month\_dec', 'month\_jul', 'month\_jun', 'month\_mar',
 'month\_may', 'month\_nov', 'month\_oct', "poutcome\_failure", "poutcome\_success"]
X=os\_data\_X[cols]
y=os\_data\_y['y']logit\_model=sm.Logit(y,X)
result=logit\_model.fit()
print(result.summary2())



Рис. 18

**Подгонка модели логистической регрессии**

from sklearn.linear\_model import LogisticRegression
from sklearn import metricsX\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)
logreg = LogisticRegression()
logreg.fit(X\_train, y\_train)



Рис. 19

**Прогнозирование результатов тестового набора и вычисление точности.**

y\_pred = logreg.predict(X\_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X\_test, y\_test)))

***Точность классификатора логистической регрессии для тестового набора: 0,74.***

**Матрица ошибок**

from sklearn.metrics import confusion\_matrix
confusion\_matrix = confusion\_matrix(y\_test, y\_pred)
print(confusion\_matrix)

***[[6124 1542]***

***[2505 5170]]***

Результат показывает, что у нас **6124+5170** верных прогнозов и **2505+1542** ошибочных.

**Вычисление точности, полноты, F-меры и поддержки**

Приведу цитату из [Scikit Learn](http://scikit-learn.org/stable/index.html%22%20%5Ct%20%22_blank):

Точность является соотношением tp/(tp + fp), где tp является числом верно-положительных, а fp — числом ложно-положительных. Точность — это интуитивно понятная способность классификатора не помечать выборку как положительную, если она отрицательна.

Полнота— это пропорция tp/(tp +fn), где tp представляет число верно-положительных результатов, а fn — число ложно-отрицательных. Полнота является интуитивно понятной способностью классификатора находить все положительные выборки.

Показатель F-бета можно интерпретировать как взвешенное гармоническое среднее точности и полноты, где лучшим значением этого показателя будет 1, а худшим 0.

Показатель F-бета определяет, насколько значимость полноты больше, чем точности, опираясь на фактор бета. Например, beta = 1.0 означает, что полнота и точность равно важны.

Поддержка — это число вхождений каждого класса в y\_test.

from sklearn.metrics import classification\_report
print(classification\_report(y\_test, y\_pred))



Рис. 20

**Интерпретация**: из всего тестового набора 74% рекламируемых срочных депозитов понравились клиентам. Из всего тестового набора 74% клиентов предпочли рекламируемые срочные депозиты.

**ROC-кривая**

from sklearn.metrics import roc\_auc\_score
from sklearn.metrics import roc\_curve
logit\_roc\_auc = roc\_auc\_score(y\_test, logreg.predict(X\_test))
fpr, tpr, thresholds = roc\_curve(y\_test, logreg.predict\_proba(X\_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit\_roc\_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log\_ROC')
plt.show()



Рис. 21

[Кривая рабочей характеристики приёмника (ROC)](https://en.wikipedia.org/wiki/Receiver_operating_characteristic) является ещё одним популярным инструментом, используемым с бинарными классификаторами. Пунктирная линия представляет ROC-кривую полностью случайного классификатора. Хороший классификатор остаётся от неё максимально далеко (по направлению к верхнему левому углу).

Jupyter notebook, использованный для написания этой статьи, доступен [здесь](https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Logistic%20Regression%20balanced.ipynb).

Читайте также:

* [7 ошибок Python, от которых стоит немедленно избавиться](https://nuancesprog.ru/p/8623/)
* [15 Python пакетов, которые нужно попробовать](https://nuancesprog.ru/p/8407/)
* [Избегайте этих нелепых ошибок при работе с Python](https://nuancesprog.ru/p/8358/)

Читайте нас в [телеграмме](https://t.me/nuancesprog), [vk](https://vk.com/nuancesprog%22%20%5Ct%20%22_blank)и [Яндекс.Дзен](https://zen.yandex.ru/profile/editor/id/5ac0d9177ddde86d7b4d096a%22%20%5Ct%20%22_blank)

*Перевод статьи*[Susan Li](https://towardsdatascience.com/%40actsusanli): [Building A Logistic Regression in Python, Step by Step.](https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8)