

Урок 1.

Введение в задачу классификации. Постановка задачи и подготовка данных.

План занятия

- [Теоретическая часть](#)
 - [Задача классификации](#)
 - [Метрики качества](#)
- [Практическая часть](#)
 - [Постановка задачи](#)
 - [Загрузка данных](#)
 - [Приведение типов](#)
 - [Обзор данных](#)
 - [Обработка пропусков](#)
 - [Обработка выбросов](#)
 - [Построение новых признаков](#)
 - [Сохранение обучающего датасета](#)
- [Курсовой проект](#)
 - [Постановка задачи](#)
 - [Примерное описание этапов выполнения курсового проекта](#)
 - [Обзор данных](#)

Теоретическая часть

Задача классификации

Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Для некоторых объектов из этого множества известна их классовая принадлежность - это подмножество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект.

Классификация объекта — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

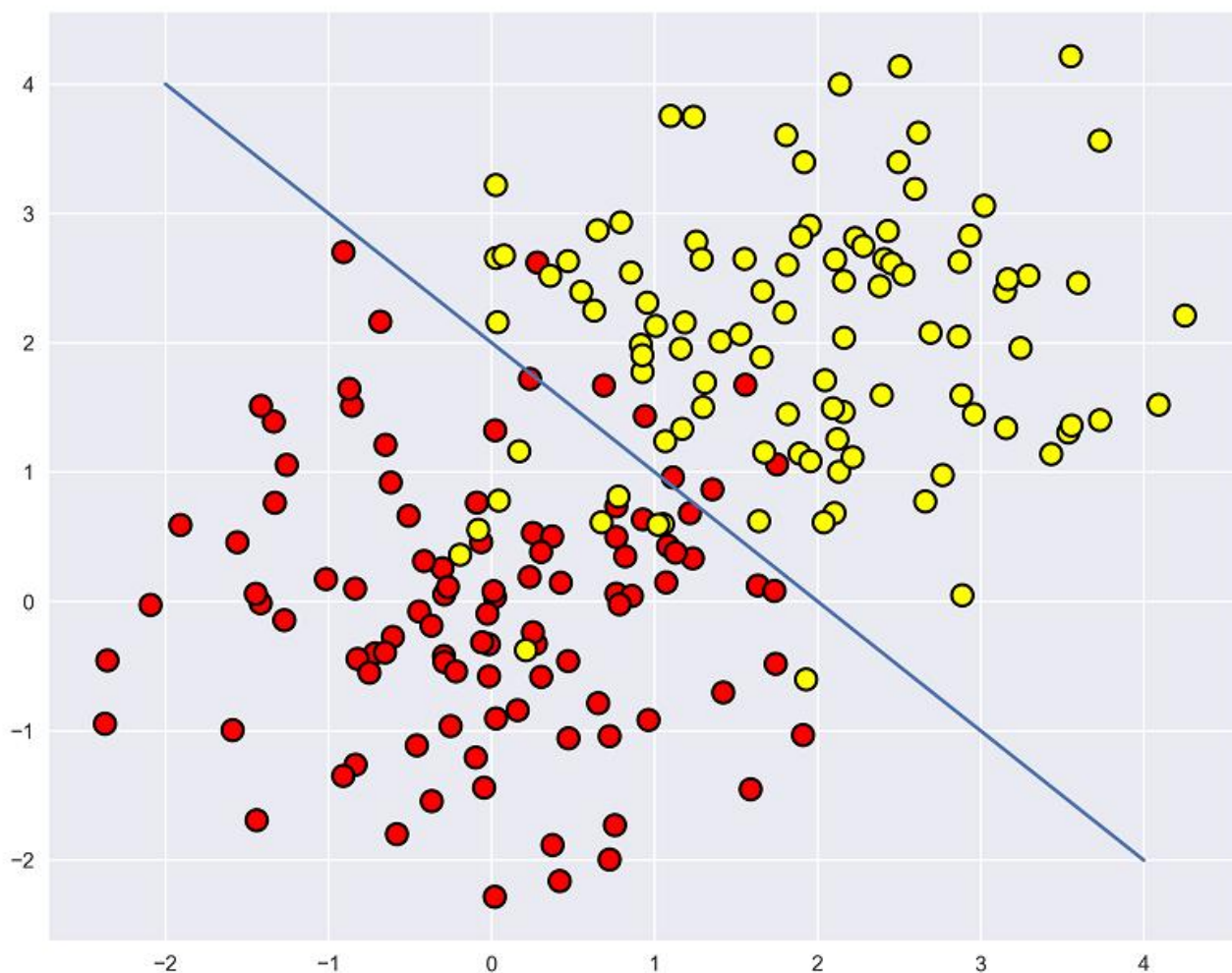
Ссылка на источник:

<http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F> (<http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F>)

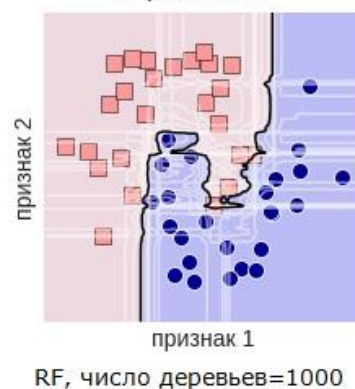
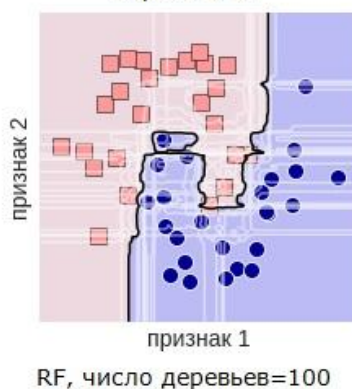
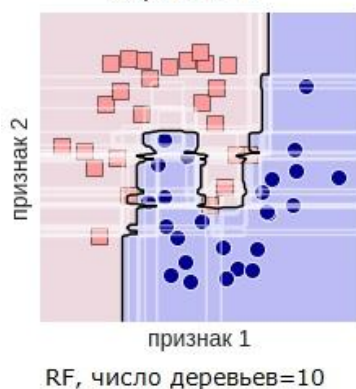
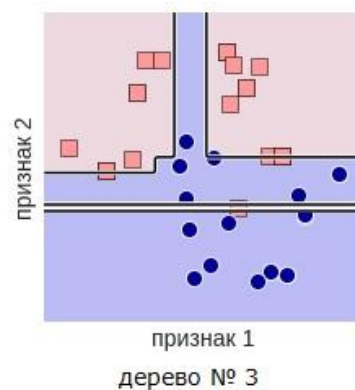
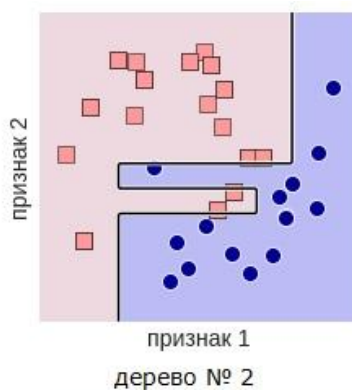
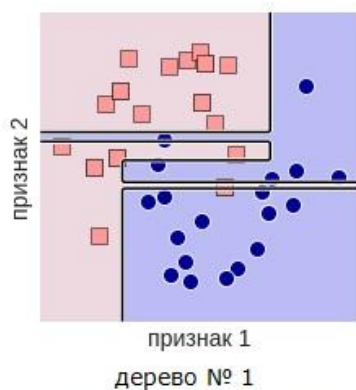
Бинарная классификация

Данные разделены на два класса, необходимо обучить модель определять принадлежность произвольного объекта (из рассматриваемого множества) к одному из них. На выходе алгоритм должен выдавать либо метку одного из двух классов, либо вероятности принадлежности рассматриваемого объекта к каждому из них.

Пример решения задачи бинарной классификации с помощью линейной модели



Пример решения задачи бинарной классификации с помощью деревьев решений

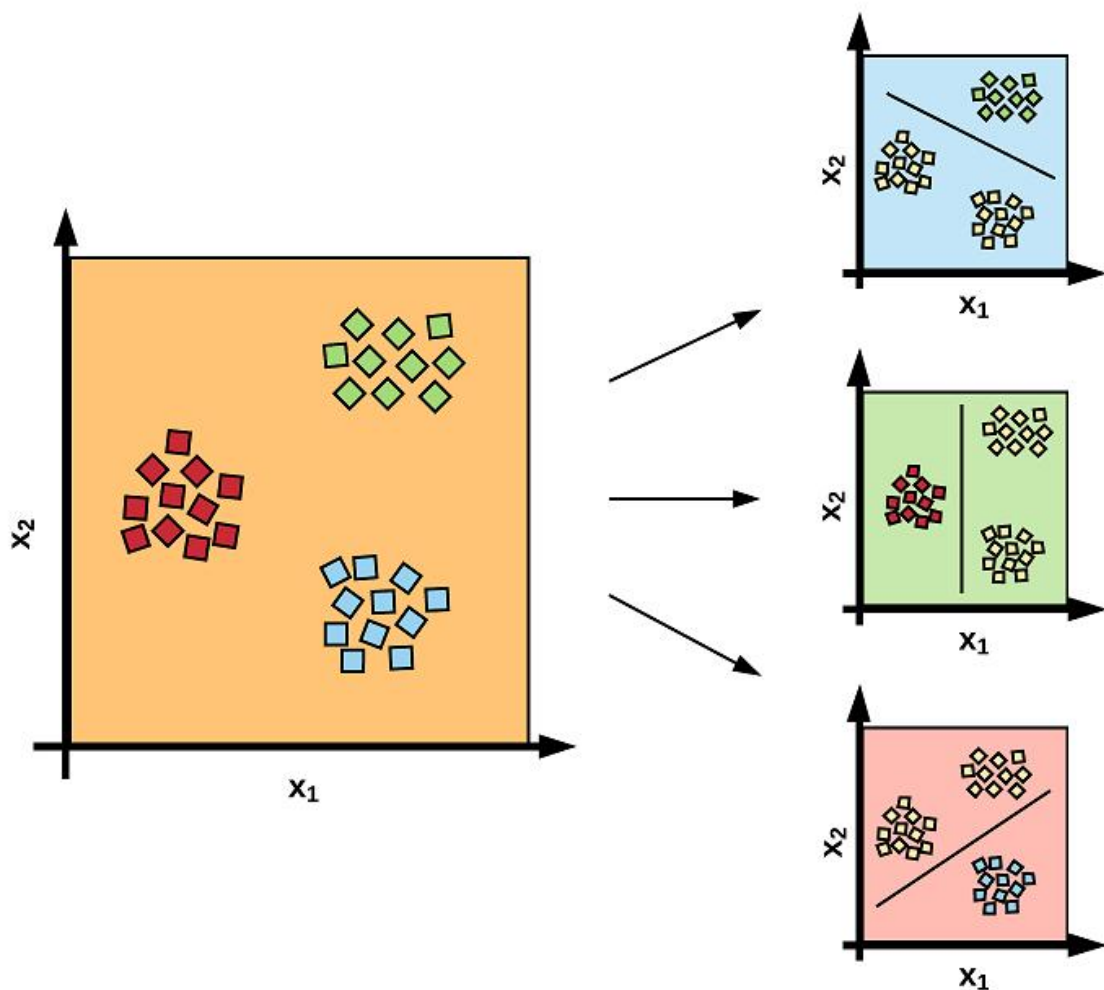


Многоклассовая классификация

Постановка задачи аналогична задаче бинарной классификации, при условии, что количество классов больше двух. Решается задача, как правило, с помощью всё тех же бинарных классификаторов, но используются различные подходы при их обучении и агрегации прогнозов.

Подход One-vs-All

Обучается N бинарных классификаторов, где N - количество классов и $N > 2$, таким образом, что каждый из них учится определять только один вид класса от всех остальных. После чего входной объект оценивается каждым из классификаторов и принимается решение об отнесении объекта к одному из N классов, исходя из того, какой классификатор показал большую степень уверенности в том, что объект относится именно к его классу.



Метрики качества

Confusion matrix (матрица ошибок)

Пример

У нас есть выборка из 13 животных, из которых 8 - кошки и 5 - другие животные. После обучения классификатора, мы попробовали определить с его помощью класс животного (кошка это или нет) и что бы более наглядно увидеть результаты его работы построили следующую таблицу, которая называется "матрицей ошибок / неточностей".

		Actual class	
		Cat	Non-cat
Predicted class	Cat	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	3 True Negatives

Positive - алгоритм предсказал, что объект относится к "основному" классу.

- True Positive (TP) - истинно-положительное решение, алгоритм предсказал, что объект относится к "основному" классу и оказался прав.
- False Positive (FP) - ложно-положительное решение, алгоритм предсказал, что объект относится к "основному" классу и ошибся.

Negative - алгоритм предсказал, что объект относится к "альтернативному" классу.

- True Negative (TN) - истинно-отрицательное решение, алгоритм предсказал, что объект относится к "альтернативному" классу и оказался прав.
- False Negative (FN) - ложно-отрицательное решение, алгоритм предсказал, что объект относится к "альтернативному" классу и ошибся.

Исходя из этой нотации, получается, что 8 животных алгоритм классифицировал верно (5 - кошек и 3 - других вида), а в 5 случаях ошибся (алгоритм не распознал 3 кошки, а 2 других вида животных наоборот принял за кошек).

Аккураси (доля правильных ответов)

Accuracy - доля правильных ответов отражает процент объектов, для которых классы были определены верно, среди всего множества объектов.

$$accuracy = \frac{T}{N} = \frac{TP + TN}{TP + TN + FP + FN}$$

T - количество объектов, для которых классы были определены верно

N - общее количество объектов

Основная проблема данной метрики в том, что она не позволяет производить более детализированную оценку работы алгоритма, как следствие она может показывать высокое качество в тех случаях, где алгоритм этого явно не заслуживает.

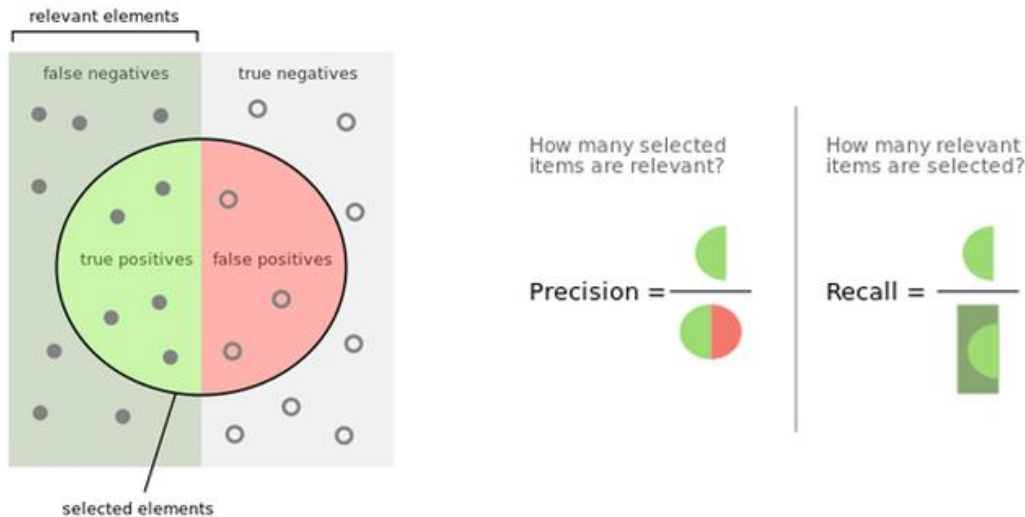
Как пример рассмотрим случай несбалансированной выборки, когда у нас есть 100 объектов, 90 из которых относятся к основному классу, а 10 к альтернативному. Мы можем добиться показателя данной метрики равного 0.9 (90% точность модели), если просто задим константную модель, которая всегда будет предсказывать основной класс.

Precision (точность) & Recall (полнота)

Precision - точность отражает какой процент объектов, отнесённых к "основному" классу, действительно являются таковыми.

Recall - полнота отражает какой процент объектов "основного" класса было обнаружено алгоритмом.

$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN}$$



F-score (F-мера, гармоническое среднее)

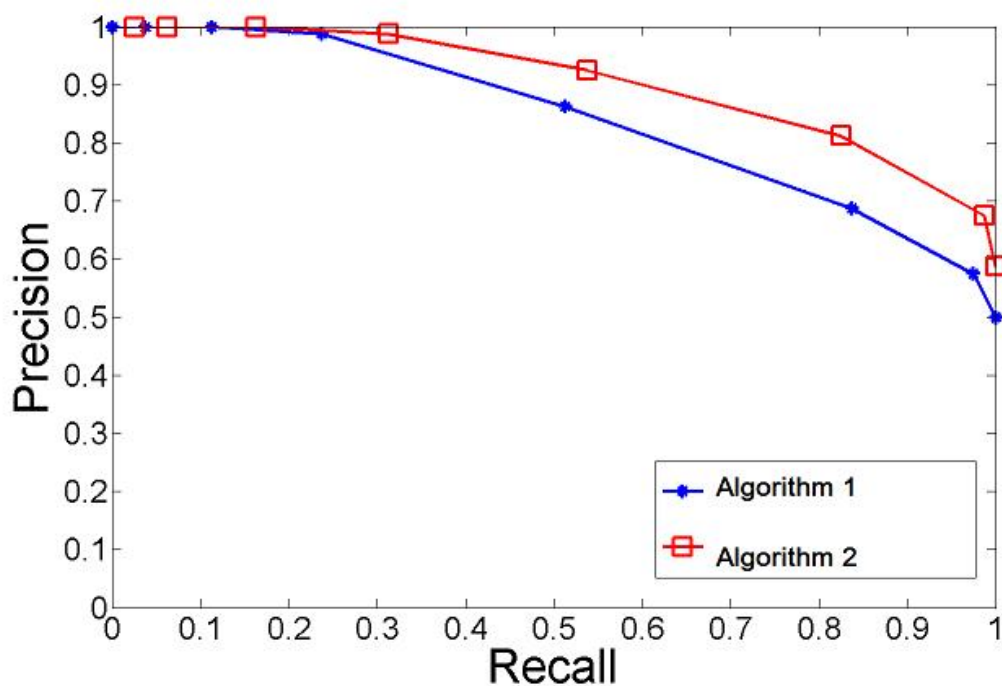
F-score - метрика объединяющая в себе точность и полноту, выдавая как результат их гармоническое среднее, а коэффициент β , позволяет балансировать вклад, вносимый точностью, тем самым акцентируя внимание на той метрике, максимизация которой для нас наиболее приоритетна.

Наиболее часто встречающаяся вариация данной метрики - это F1-score, когда $\beta = 1$, а влияние точности и полноты одинаково.

$$F_{\beta} = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

PR-кривая

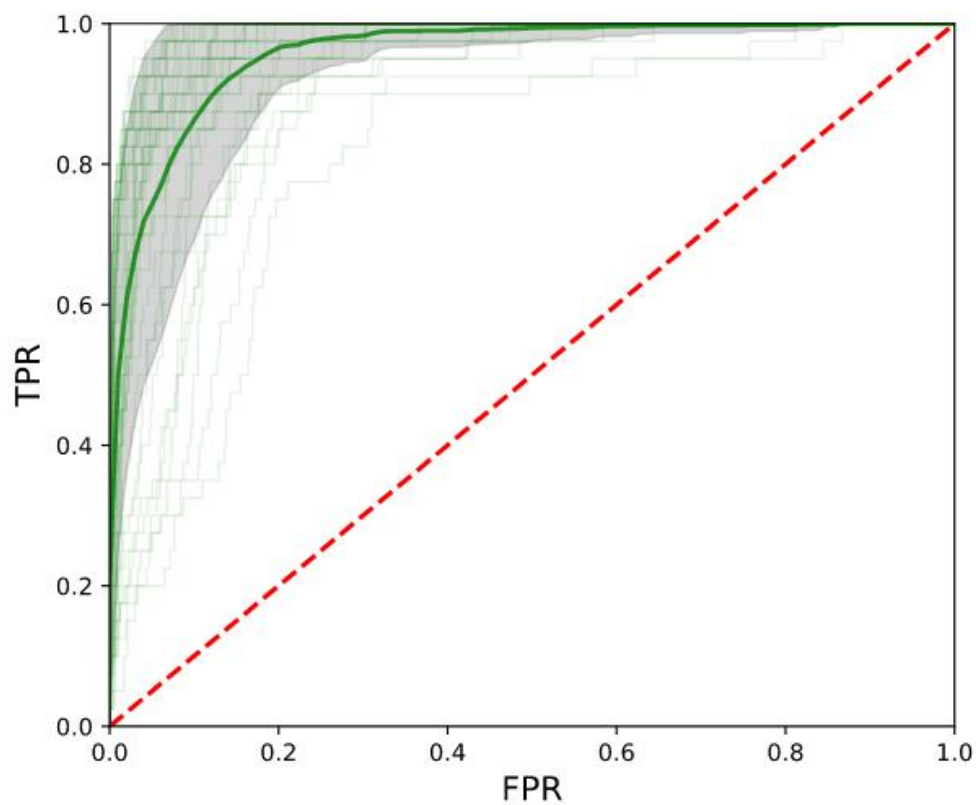
Графическая характеристика качества бинарного классификатора, отображает зависимость изменения точности алгоритма от его полноты.



ROC-кривая

Графическая характеристика качества бинарного классификатора, отображает зависимость доли верных положительных классификаций *True Positive Rate (TPR)* от доли ложных положительных классификаций *False Positive Rate (FPR)*.

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN}$$



Практическая часть

Постановка задачи

Требуется на основании демографических данных о клиентах банка, истории их платежей и состоянии счета, предсказать невыполнение их обязательств по кредитной карте в следующем месяце.

Описание датасета

- **LIMIT_BAL** - Сумма предоставленного кредита
- **SEX** - Пол (1=мужчина, 2=женщина)
- **EDUCATION** - Образование (1=аспирантура, 2=университет, 3=старшая школа, 4=прочее, 5=неизвестно, 6=неизвестно)
- **MARRIAGE** - Семейное положение (1=женат/замужен, 2=не женат/не замужен, 3=прочее)
- **AGE** - Возраст (в годах)
- **PAY_1** - Статус погашения в Сентябре (-1=погашен полностью, 0=погашен частично, 1=отсрочка платежа на один месяц, ..., 3=отсрочка платежа на три месяца и более)
- **PAY_2** - Статус погашения в Августе
- **PAY_3** - Статус погашения в Июле
- **PAY_4** - Статус погашения в Июне
- **PAY_5** - Статус погашения в Мае
- **PAY_6** - Статус погашения в Апреле
- **BILL_AMT1** - Сумма выписки по счету в Сентябре
- **BILL_AMT2** - Сумма выписки по счету в Августе
- **BILL_AMT3** - Сумма выписки по счету в Июле
- **BILL_AMT4** - Сумма выписки по счету в Июне
- **BILL_AMT5** - Сумма выписки по счету в Мае
- **BILL_AMT6** - Сумма выписки по счету в Апреле
- **PAY_AMT1** - Сумма предыдущего платежа в Сентябре
- **PAY_AMT2** - Сумма предыдущего платежа в Августе
- **PAY_AMT3** - Сумма предыдущего платежа в Июле
- **PAY_AMT4** - Сумма предыдущего платежа в Июне
- **PAY_AMT5** - Сумма предыдущего платежа в Мае
- **PAY_AMT6** - Сумма предыдущего платежа в Апреле
- **NEXT_MONTH_DEFAULT** - Просрочка платежа в следующем месяце (1=да, 0=нет)

Подключение библиотек и скриптов

```
In [1]: import numpy as np
import pandas as pd
```

Пути к директориям и файлам

```
In [2]: # input
DATASET_PATH = '../training_project_data.csv'

# output
PREP_DATASET_PATH = '../training_project_data_prep.csv'
```

Загрузка данных


```
In [3]: df = pd.read_csv(DATASET_PATH)
df.head()
```

```
Out[3]:
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AM
0	150000.0	2	2	2	24	1	2	0	0	0	...	17853.0	3452
1	50000.0	2	3	1	46	3	3	3	3	2	...	30748.0	3016
2	150000.0	2	2	1	41	-1	-1	-1	-1	0	...	6784.0	692
3	150000.0	2	2	2	35	0	0	0	0	0	...	55182.0	4898
4	70000.0	2	1	1	35	1	2	2	2	2	...	47026.0	4597

5 rows × 24 columns

```
In [4]: df.shape
```

```
Out[4]: (10000, 24)
```

```
In [5]: df.iloc[0]
```

```
Out[5]: LIMIT_BAL      150000.0
SEX                2.0
EDUCATION          2.0
MARRIAGE           2.0
AGE                24.0
PAY_1              1.0
PAY_2              2.0
PAY_3              0.0
PAY_4              0.0
PAY_5              0.0
PAY_6              0.0
BILL_AMT1          65542.0
BILL_AMT2          28311.0
BILL_AMT3          23978.0
BILL_AMT4          17853.0
BILL_AMT5          34527.0
BILL_AMT6          28277.0
PAY_AMT1           0.0
PAY_AMT2           2145.0
PAY_AMT3          10000.0
PAY_AMT4          22000.0
PAY_AMT5           3000.0
PAY_AMT6           5093.0
NEXT_MONTH_DEFAULT 0.0
Name: 0, dtype: float64
```



```
In [6]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 24 columns):
LIMIT_BAL          10000 non-null float64
SEX                10000 non-null int64
EDUCATION          10000 non-null int64
MARRIAGE           10000 non-null int64
AGE               10000 non-null int64
PAY_1              10000 non-null int64
PAY_2              10000 non-null int64
PAY_3              10000 non-null int64
PAY_4              10000 non-null int64
PAY_5              10000 non-null int64
PAY_6              10000 non-null int64
BILL_AMT1          10000 non-null float64
BILL_AMT2          10000 non-null float64
BILL_AMT3          10000 non-null float64
BILL_AMT4          10000 non-null float64
BILL_AMT5          10000 non-null float64
BILL_AMT6          10000 non-null float64
PAY_AMT1           10000 non-null float64
PAY_AMT2           10000 non-null float64
PAY_AMT3           10000 non-null float64
PAY_AMT4           10000 non-null float64
PAY_AMT5           10000 non-null float64
PAY_AMT6           10000 non-null float64
NEXT_MONTH_DEFAULT 10000 non-null int64
dtypes: float64(13), int64(11)
memory usage: 1.8 MB
```

Приведение типов

```
In [7]: for colname in ['SEX', 'EDUCATION', 'MARRIAGE', 'PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']:
        df[colname] = df[colname].astype(str)
```

```
In [8]: df.dtypes
```

```
Out[8]: LIMIT_BAL          float64
SEX                object
EDUCATION          object
MARRIAGE           object
AGE               int64
PAY_1              object
PAY_2              object
PAY_3              object
PAY_4              object
PAY_5              object
PAY_6              object
BILL_AMT1          float64
BILL_AMT2          float64
BILL_AMT3          float64
BILL_AMT4          float64
BILL_AMT5          float64
BILL_AMT6          float64
PAY_AMT1           float64
PAY_AMT2           float64
PAY_AMT3           float64
PAY_AMT4           float64
PAY_AMT5           float64
PAY_AMT6           float64
NEXT_MONTH_DEFAULT int64
dtype: object
```

Обзор данных

Обзор целевой переменной

```
In [9]: df['NEXT_MONTH_DEFAULT'].value_counts()
```

```
Out[9]: 0      7805  
        1      2195  
        Name: NEXT_MONTH_DEFAULT, dtype: int64
```

Обзор количественных признаков

```
In [10]: df.describe()
```

```
Out[10]:
```

	LIMIT_BAL	AGE	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	
count	10000.000000	10000.00000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	1
mean	168182.000000	35.39600	51490.560500	49298.972000	47292.40850	43605.665800	40347.829700	3
std	130118.668771	9.19536	74906.094564	71808.952137	69398.13263	65062.788893	61044.676044	5
min	10000.000000	21.00000	-11545.000000	-24704.000000	-157264.00000	-65167.000000	-23003.000000	-33
25%	50000.000000	28.00000	3388.000000	3016.500000	2561.75000	2400.000000	1703.250000	
50%	140000.000000	34.00000	22896.500000	21576.000000	20370.50000	19194.000000	18261.000000	1
75%	240000.000000	41.00000	67834.750000	64243.500000	60783.25000	55731.250000	50491.000000	4
max	1000000.000000	75.00000	964511.000000	983931.000000	689643.00000	891586.000000	927171.000000	96

Обзор номинативных признаков

```
In [11]: for cat_colname in df.select_dtypes(include='object').columns:
          print(str(cat_colname) + '\n\n' + str(df[cat_colname].value_counts()) + '\n' + '*'
          * 100 + '\n')
```

SEX

2 6068

1 3932

Name: SEX, dtype: int64

EDUCATION

2 4678

1 3496

3 1674

5 90

4 42

6 18

0 2

Name: EDUCATION, dtype: int64

MARRIAGE

2 5332

1 4547

3 108

0 13

Name: MARRIAGE, dtype: int64

PAY_1

0 4895

-1 2852

1 1217

2 881

3 155

Name: PAY_1, dtype: int64

PAY_2

0 5178

-1 3318

2 1318

3 175

1 11

Name: PAY_2, dtype: int64

PAY_3

0 5200

-1 3379

2 1288

3 131

1 2

Name: PAY_3, dtype: int64

PAY_4

0 5459

-1 3360

2 1064

3 117

Name: PAY_4, dtype: int64

PAY_5

```
0      5601
-1     3398
2       894
3       107
```

Name: PAY_5, dtype: int64

```
*****
*****
```

PAY_6

```
0      5394
-1     3585
2       927
3        94
```

Name: PAY_6, dtype: int64

```
*****
*****
```

Обработка пропусков

```
In [12]: len(df) - df.count()
```

```
Out[12]: LIMIT_BAL      0
SEX      0
EDUCATION  0
MARRIAGE  0
AGE      0
PAY_1     0
PAY_2     0
PAY_3     0
PAY_4     0
PAY_5     0
PAY_6     0
BILL_AMT1  0
BILL_AMT2  0
BILL_AMT3  0
BILL_AMT4  0
BILL_AMT5  0
BILL_AMT6  0
PAY_AMT1   0
PAY_AMT2   0
PAY_AMT3   0
PAY_AMT4   0
PAY_AMT5   0
PAY_AMT6   0
NEXT_MONTH_DEFAULT  0
dtype: int64
```

Обработка выбросов

EDUCATION

```
In [13]: df.loc[df['EDUCATION'] == '0', 'EDUCATION'] = df['EDUCATION'].mode()[0]
```

MARRIAGE

```
In [14]: df.loc[df['MARRIAGE'] == '0', 'MARRIAGE'] = df['MARRIAGE'].mode()[0]
```

Построение новых признаков

Id

```
In [15]: df['ID'] = df.index.tolist()
```

Dummies

```
In [16]: df['IS_MALE'] = df['SEX'].map({'1':'1', '2':'0'}).astype(int)
```

```
In [17]: for cat_colname in df.select_dtypes(include='object').columns[1:]:  
         df = pd.concat([df, pd.get_dummies(df[cat_colname], prefix=cat_colname)], axis=1)
```

Сохранение обучающего датасета

```
In [18]: df.to_csv(PREP_DATASET_PATH, index=False, encoding='utf-8')
```

Курсовой проект

Постановка задачи

Задача

Требуется, на основании имеющихся данных о клиентах банка, построить модель, используя обучающий датасет, для прогнозирования невыполнения долговых обязательств по текущему кредиту. Выполнить прогноз для примеров из тестового датасета.

Наименование файлов с данными

course_project_train.csv - обучающий датасет

course_project_test.csv - тестовый датасет

Целевая переменная

Credit Default - факт невыполнения кредитных обязательств

Метрика качества

F1-score (sklearn.metrics.f1_score)

Требования к решению

Целевая метрика

- $F1 > 0.5$
- Метрика оценивается по качеству прогноза для главного класса (1 - просрочка по кредиту)

Решение должно содержать

1. Тетрадка Jupyter Notebook с кодом Вашего решения, названная по образцу {ФИО}_solution.ipynb, пример SShirkin_solution.ipynb
2. Файл CSV с прогнозами целевой переменной для тестового датасета, названный по образцу {ФИО}_predictions.csv, пример SShirkin_predictions.csv

Рекомендации для файла с кодом (ipynb)

1. Файл должен содержать заголовки и комментарии (markdown)
2. Повторяющиеся операции лучше оформлять в виде функций
3. Не делать вывод большого количества строк таблиц (5-10 достаточно)
4. По возможности добавлять графики, описывающие данные (около 3-5)
5. Добавлять только лучшую модель, то есть не включать в код все варианты решения проекта
6. Скрипт проекта должен отрабатывать от начала и до конца (от загрузки данных до выгрузки предсказаний)
7. Весь проект должен быть в одном скрипте (файл ipynb).
8. Допускается применение библиотек Python и моделей машинного обучения, которые были в данном курсе.

Сроки сдачи

Сдать проект нужно в течение 5 дней после окончания последнего вебинара. Оценки работ, сданных до дедлайна, будут представлены в виде рейтинга, ранжированного по заданной метрике качества. Проекты, сданные после дедлайна или сданные повторно, не попадают в рейтинг, но можно будет узнать результат.

Примерное описание этапов выполнения курсового проекта

Построение модели классификации

1. Обзор обучающего датасета
2. Обработка выбросов
3. Обработка пропусков
4. Анализ данных
5. Отбор признаков
6. Балансировка классов
7. Подбор моделей, получение бейзлана
8. Выбор наилучшей модели, настройка гиперпараметров
9. Проверка качества, борьба с переобучением
10. Интерпретация результатов

Прогнозирование на тестовом датасете

1. Выполнить для тестового датасета те же этапы обработки и построения признаков
2. Спрогнозировать целевую переменную, используя модель, построенную на обучающем датасете
3. Прогнозы должны быть для всех примеров из тестового датасета (для всех строк)
4. Соблюдать исходный порядок примеров из тестового датасета

Обзор данных

Описание датасета

- **Home Ownership** - домовладение
- **Annual Income** - годовой доход
- **Years in current job** - количество лет на текущем месте работы
- **Tax Liens** - налоговые обременения
- **Number of Open Accounts** - количество открытых счетов
- **Years of Credit History** - количество лет кредитной истории
- **Maximum Open Credit** - наибольший открытый кредит
- **Number of Credit Problems** - количество проблем с кредитом
- **Months since last delinquent** - количество месяцев с последней просрочки платежа
- **Bankruptcies** - банкротства
- **Purpose** - цель кредита
- **Term** - срок кредита
- **Current Loan Amount** - текущая сумма кредита
- **Current Credit Balance** - текущий кредитный баланс
- **Monthly Debt** - ежемесячный долг
- **Credit Default** - факт невыполнения кредитных обязательств (0 - погашен вовремя, 1 - просрочка)

Пути к директориям и файлам

```
In [19]: TRAIN_DATASET_PATH = '../course_project_train.csv'
         TEST_DATASET_PATH = '../course_project_test.csv'
```

Загрузка данных

```
In [20]: df_train = pd.read_csv(TRAIN_DATASET_PATH)
df_train.head()
```

Out [20]:

	Home Ownership	Annual Income	Years in current job	Tax Liens	Number of Open Accounts	Years of Credit History	Maximum Open Credit	Number of Credit Problems	Months since last delinquent	Bankruptcies	Purp
0	Home Mortgage	1025981.0	7 years	0.0	8.0	13.1	431640.0	0.0	NaN	0.0	consolid
1	Rent	595726.0	10+ years	0.0	14.0	15.4	493702.0	1.0	NaN	1.0	consolid
2	Home Mortgage	1266882.0	10+ years	0.0	15.0	26.2	477004.0	0.0	11.0	0.0	consolid
3	Rent	NaN	10+ years	0.0	9.0	16.0	236742.0	0.0	NaN	0.0	consolid
4	Home Mortgage	685767.0	2 years	0.0	10.0	25.4	389026.0	2.0	61.0	2.0	consolid

```
In [21]: df_train.shape
```

Out [21]: (7500, 16)

```
In [22]: df_train.iloc[0]
```

```
Out [22]: Home Ownership          Home Mortgage
Annual Income          1.02598e+06
Years in current job          7 years
Tax Liens                0
Number of Open Accounts      8
Years of Credit History      13.1
Maximum Open Credit        431640
Number of Credit Problems    0
Months since last delinquent  NaN
Bankruptcies                0
Purpose                    debt consolidation
Term                      Short Term
Current Loan Amount          1e+08
Current Credit Balance      144267
Monthly Debt                19152
Credit Default              0
Name: 0, dtype: object
```

```
In [23]: df_train['Credit Default'].value_counts()
```

```
Out [23]: 0    5793
1    1707
Name: Credit Default, dtype: int64
```

```
In [24]: df_test = pd.read_csv(TEST_DATASET_PATH)
df_test.head()
```

Out [24]:

	Home Ownership	Annual Income	Years in current job	Tax Liens	Number of Open Accounts	Years of Credit History	Maximum Open Credit	Number of Credit Problems	Months since last delinquent	Bankruptcies	Purp
0	Home Mortgage	930620.0	1 year	0.0	13.0	11.1	361812.0	1.0	NaN	1.0	consolid
1	Home Mortgage	1592789.0	< 1 year	0.0	4.0	17.1	414018.0	0.0	NaN	0.0	
2	Rent	NaN	NaN	1.0	12.0	24.6	275704.0	2.0	NaN	1.0	consolid
3	Home Mortgage	1104185.0	10+ years	0.0	10.0	20.0	1023440.0	0.0	NaN	0.0	consolid
4	Home Mortgage	1852823.0	6 years	0.0	10.0	26.6	1771880.0	0.0	13.0	0.0	consolid

```
In [25]: df_test.shape
```

```
Out[25]: (2500, 15)
```

```
In [ ]:
```