|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования Российской Федерации  Федеральное государственное бюджетное образовательное учреждение высшего  образования  **6.jpg«Московский авиационный институт (национальный исследовательский университет)»** | | |
| **mai.gif** | **Кафедра 609**  **«Прикладная информатика»** |  |

**ОТЧЁТ**

**О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Институт №6 «Аэрокосмический»

направления подготовки

09.03.01. «Информатика и вычислительная техника»

Принял: д. т. н., профессор

Кондрашов Юрий Николаевич /\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_» \_\_\_\_\_\_\_\_ 2020 г.

*(подпись преподавателя)*

Выполнила студентка группы М60-308Б-17

Сёмина Мария Вячеславовна /\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_» \_\_\_\_\_\_\_\_ 2020 г.

*(фамилия, имя, отчество)* *(подпись студента) (дата)*

**Цели:**

* Исследование возможностей библиотек pandas, seasborn для вывода статистики
* Исследование алгоритмов классификации на примере библиотеки sklearn
* Сравнение алгоритмов классификации для кредитного скоринга.

**Исходные данные:**

Для построения моделей в данной работе используется широко известный Немецкий Набор Данных (German Credit Data Set). Он содержит 1 000 объектов – анкеты людей, желавших получить кредит в банке, из которых 700 состоятельны и 300 несостоятельны. Описание заемщика содержит 20 атрибутов. Немецкий набор данных находится в публичном доступе в UCI Repository of Machine Learning.

Признаками заёмщика являются:

1) existingchecking – наличие у клиента средств на текущем счёте.

A11: < 0 денежных единиц

A12: 0 <= ... <200 денежных единиц

A13: ...> = 200 денежных единиц

A14: нет текущего счета

2) duration – длительность запрашиваемого кредита в месяцах

3) credithistory – кредитная история

A30: кредиты не получены / все кредиты возвращены должным образом

A31: все кредиты в этом банке погашены должным образом

A32: существующие кредиты возвращены должным образом на данный момент

A33: задержка погашения в прошлом

A34: текущий счет с просрочкой / другие кредиты (не в этом банке)

4) purpose – цель кредита.

A40: автомобиль (новый)

A41: автомобиль (б/у)

A42: мебель / оборудование

A43: радио / телевидение

A44: бытовая техника (товары для дома)

A45: ремонт

A46: образование

А48: переподготовка

A49: бизнес

A410: другие

5) creditamount – сумма кредита

6) savings – сбережения на сберегательном счёте

A61: ... <100 денежных единиц

A62: 100 <= ... <500 денежных единиц

A63: 500 <= ... <1000 денежных единиц

A64: > = 1000 денежных единиц

A65: неизвестно / сберегательный счет отсутствует

7) employmentsince – количество лет, в течение которой клиент работает на текущей работе

A71: безработный

A72: ... < 1 года

A73: 1 <= ... < 4 лет

A74: 4 <= ... < 7 лет

A75: >= 7 лет

8) installmentrate – процентная ставка как процент предполагаемого дохода

9) statussex – пол и семейное положение

A91: мужчина: разведенный

A92: женщина: разведена / замужем

A93: мужчина: холост

A94: мужчина: женат / овдовел

A95: женщина: не замужем

10) otherdebtors – созаёмщики

A101 : нет

A102 : есть созаёмщик

A103 : есть поручитель

11) residencesince – время, в течение которого клиент живёт в текущем месте жительства

12) property – ценное имущество

A121: недвижимость

A122: если нет A121: договор о сбережении строительного общества / страхование жизни

A123: если не A121 / A122: машина или другое, из не перечисленного

A124: неизвестно / нет собственности

13) age - возраст

14) otherinstallmentplans – наличие других обязательств (рассрочек)

A141: банк

A142: обязательство перед магазинами

A143: нет

15) housing – тип жилья

A151: съёмное

A152: своё

A153: социальный найм/другое

16) existingcredits – количество открытых кредитов в этом банке

17) job – тип работы

A171: безработный/ иностранный гражданин низкой квалификации

A172: низкоквалифицированный гражданин страны банка

A173: квалифицированный работник

A174: менеджмент/высококвалифицированный работник/самозанятый

18) peopleliable – количество людей, за которых отвечает заёмщик

19) telephone – наличие телефона

A191: нет

A192: есть, зарегистрирован на имя клиента

20) foreignworker – является ли клиент зарубежным работником

A201: да

A202: нет

21) classification – целевая переменная.

1 – клиент вернул кредит

2 – клиент не вернул кредит

# Анализ исходных данных

Сначала загружаем необходимые библиотеки:

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

import numpy as np ; np.random.seed(sum(map(ord, "aesthetics")))

from sklearn.datasets import make\_classification

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import classification\_report as cr

from sklearn.metrics import classification\_report,confusion\_matrix, roc\_curve, roc\_auc\_score, auc, accuracy\_score

from sklearn.model\_selection import ShuffleSplit,train\_test\_split, cross\_val\_score, GridSearchCV

from sklearn.preprocessing import OneHotEncoder, LabelEncoder, label\_binarize, StandardScaler, MinMaxScaler

import warnings

warnings.filterwarnings("ignore")

from pprint import pprint

from collections import defaultdict

Далее загрузим в pandas.DataFrame – объект с именем data таблицу с данными и выведем первые её пять строк:

data = pd.read\_csv("german\_data.csv", sep = ';')

data.head()

Результат выполнения:

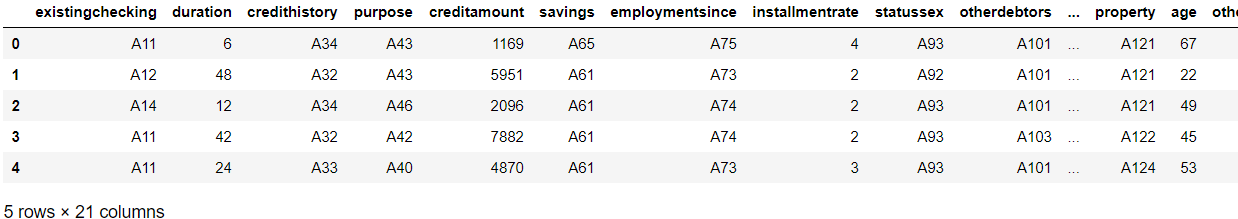


Рисунок Первые 5 строк исходных данных

Выведем краткую информацию о данных:

data.info()

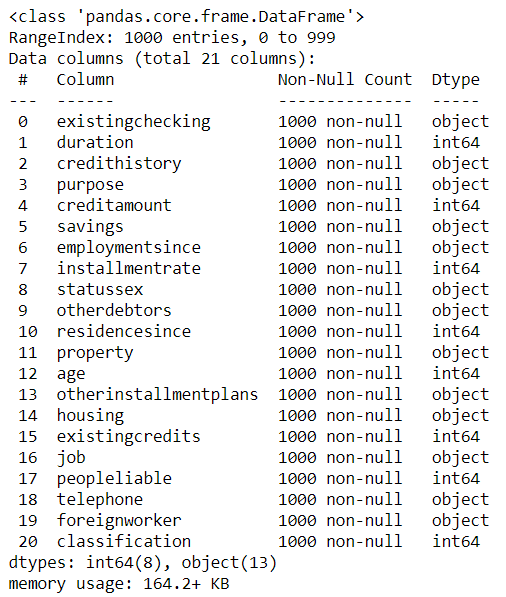


Рисунок Краткая информация о данных

В данных нет пропущенных значений, а также большинство признаков не являются числовыми и требуют преобразования, т.к. многие алгоритмы работают только с числовыми признаками.

Значения целевой переменной classification преобразуем из {1;2} на {0;1} с помощью следующего кода:

data.classification.replace([1,2], [1,0], inplace=True)

Числовые признаки нормализуем:

numvars = [ 'duration', 'creditamount', 'installmentrate', 'residencesince', 'age',

'existingcredits', 'peopleliable', 'classification']

numdata\_std=pd.DataFrame (StandardScaler().fit\_transform(data[numvars].drop(['classification'], axis=1)))

Далее с помощью LabelEncoder перекодируем категориальные переменные (приведём к виду [0, 1, 2, 3, … ] ). После этого небинарные переменные превращаем в несколько столбцов. Выведем признаки до и после преобразования:

catvars = ['existingchecking', 'credithistory', 'purpose', 'savings', 'employmentsince',

'statussex', 'otherdebtors', 'property', 'otherinstallmentplans', 'housing', 'job',

'telephone', 'foreignworker']

d = defaultdict(LabelEncoder)

#Перекодирование

ndata = data[catvars].apply(lambda x: d[x.name].fit\_transform(x))

#Вывод преобразованных переменных

for x in range(len(catvars)):

print(catvars[x],": ", data[catvars[x]].unique())

print(catvars[x],": ", ndata[catvars[x]].unique())

dummyvars = pd.get\_dummies(data[catvars])

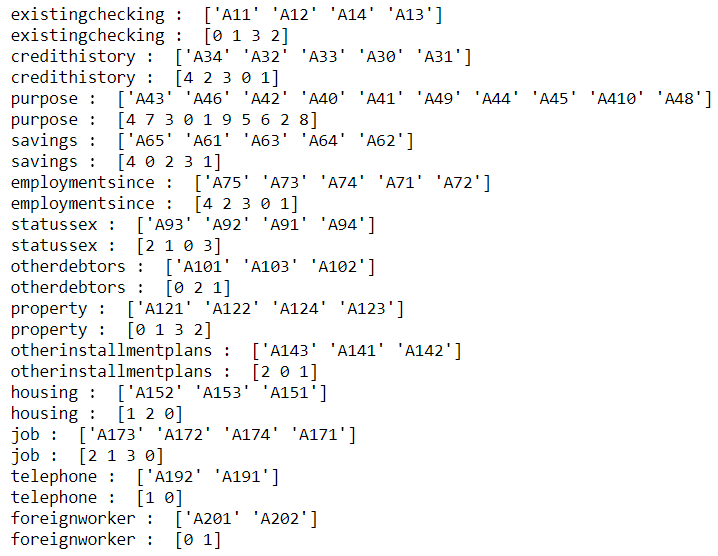


Рисунок Перекодирование категориальных переменных

Получаем новую таблицу с преобразованными данными, выводим её размер:

data\_new = pd.concat([data[numvars], dummyvars], axis = 1)

dc = pd.concat([ndata, data[numvars]], axis = 1)

print(data\_new.shape)

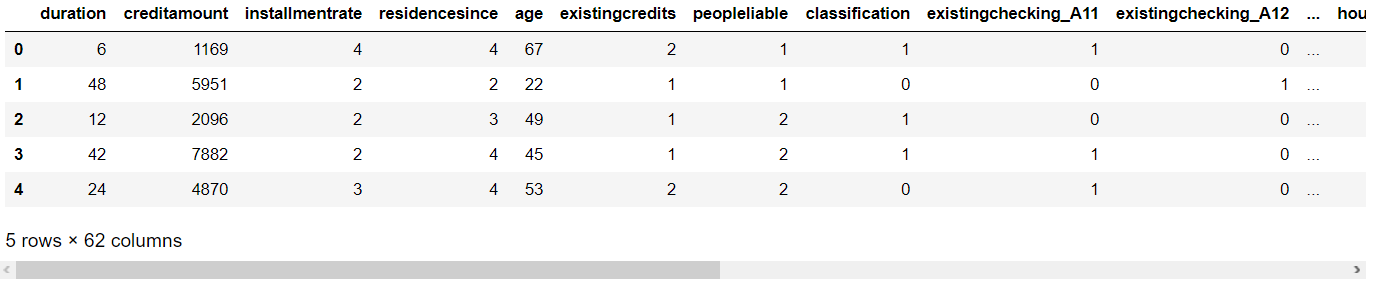


Рисунок Преобразованные данные

Количество столбцов увеличилось на 21 из-за преобразования категориальных переменных в бинарные.

Выведем диаграмму рассеяния зависимости суммы кредита от возраста:

plt.scatter(data\_new['creditamount'],data\_new["age"])

plt.xlabel('Credit amount', size=16); plt.ylabel('Age', size=16); plt.title('Distribution of Credit amount', size=16)

plt.tick\_params(axis='both', which='major', labelsize=14)

plt.figure()

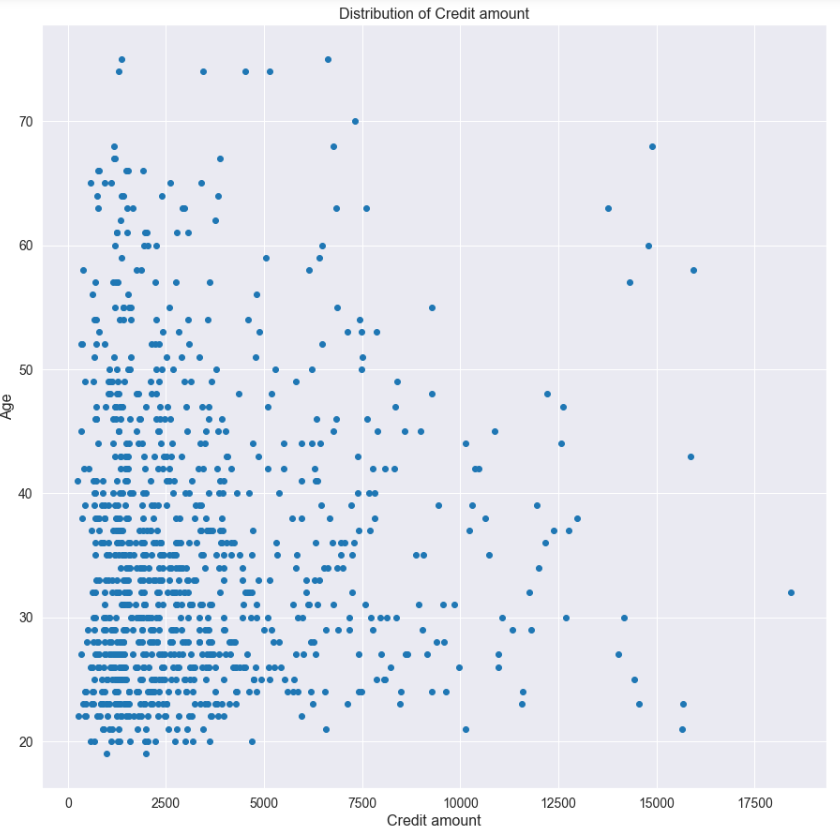


Рисунок Диаграмма рассеяния зависимости суммы кредита от возраста

Наибольшее количество кредитов в левом нижнем углу – т.е. это молодые люди и люди среднего возраста с относительно небольшими суммами кредита.

Далее покажем распределение возраста в зависимости от того, вернул клиент кредит или нет – для этого построим KDE-график (kernel density estimation) – распределения ядерной плотности.

#KDE - график займов

sns.kdeplot(data.loc[data['classification'] == 1, 'age'], label = 'Вернул')

sns.kdeplot(data.loc[data['classification'] == 0, 'age'], label = 'Не вернул')

plt.rcParams['figure.figsize'] = (15,15)

plt.xlabel('Age (years)')

plt.ylabel('Density')

plt.title('Distribution of Ages')

plt.figure(figsize = (20, 20))

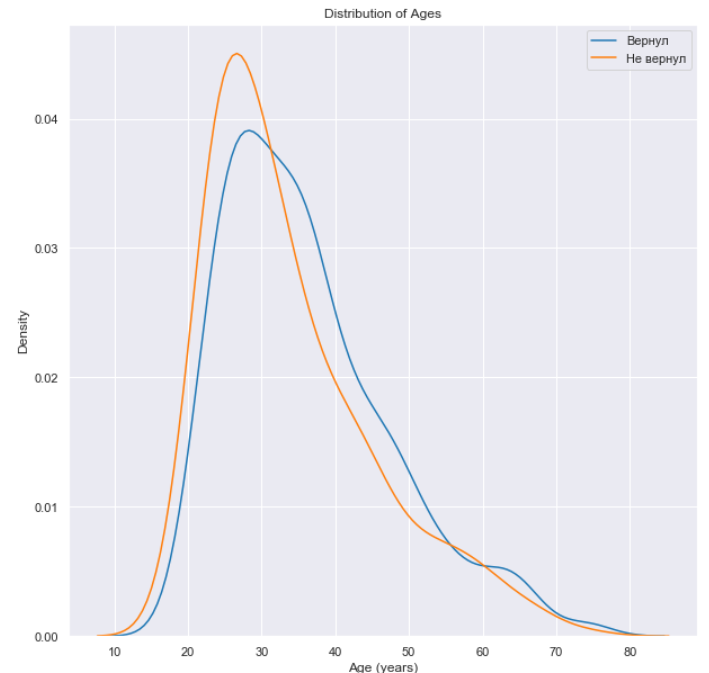


Рисунок KDE-график возраста

Видно, что доля невозврата среди людей 20-30 больше всего.

Выведем на box-plot длительность займа:

plt.title('Boxplot:Duration')

plt.ylabel('Duration(Month)')

sns.boxplot(data=data.duration, width=.3 ,linewidth=2.5, color='m')

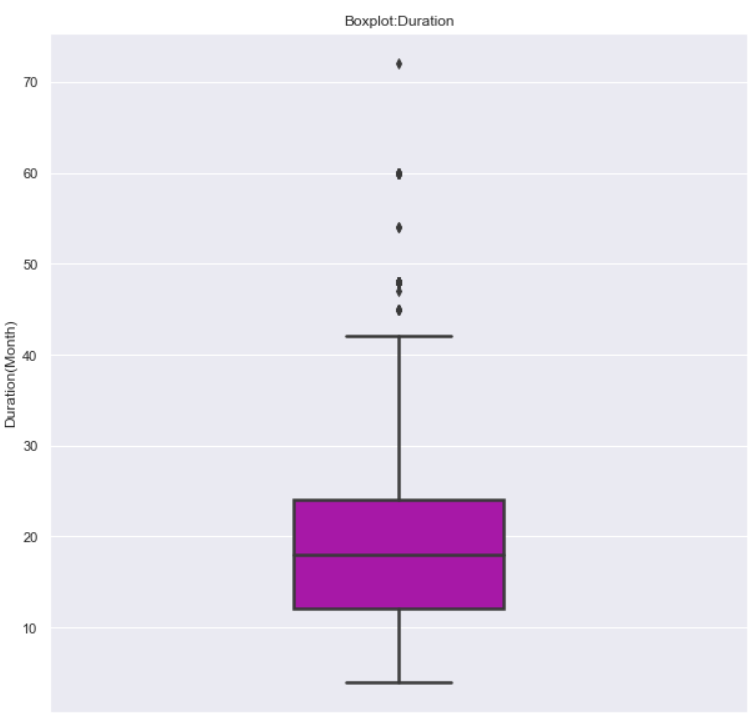


Рисунок Длительность займа

Т.е. в среднем клиенты берут кредит примерно на 17 месяцев.

Выведем статистику по отдельным признакам.

Для вывода статистики будет использоваться следующая функция:

def plt\_stat(var):

temp = dc[var].value\_counts()

df1 = pd.DataFrame({var: temp.index,'Количество займов': temp.values})

class\_perc = dc[[var, 'classification']].groupby([var],as\_index=False).mean()

class\_perc.sort\_values(by='classification', ascending=False, inplace=True)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(8,6))

sns.barplot(ax=ax1, x = var, y="Количество займов",data=df1, palette="deep")

sns.barplot(ax=ax2, x = var, y="classification", data=class\_perc, palette="deep")

plt.ylabel("Доля проблемных", fontsize=10)

plt.tick\_params(axis='both', which='major', labelsize=10)

plt.show();

Функция выводит 2 графика – слева распределение клиентов по заданному признаку, справа – клиенты, которые не вернули кредиты.

Выведем с помощью функции статистику для цели кредита (purpose):

plt\_stat('purpose')

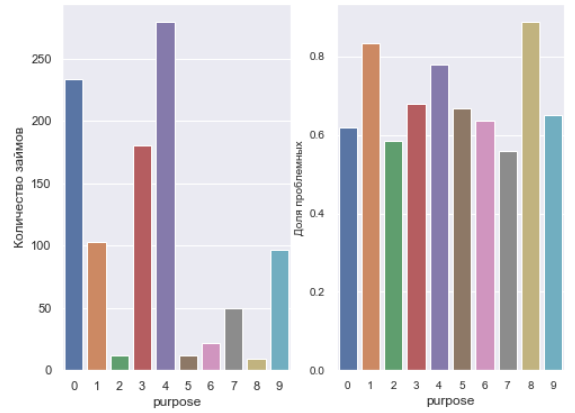


Рисунок Количество кредитов в зависимости от цели

Причин покупки 2, 5 и 8 (переобучение, товары для дома и другое) не так много в выборке (менее 25 займов), однако их достаточно много в тех случаях, когда клиент не вернул кредит (особенно по причине «другое»).

Выведем статистику для кредитной истории:

plt\_stat('credithistory')

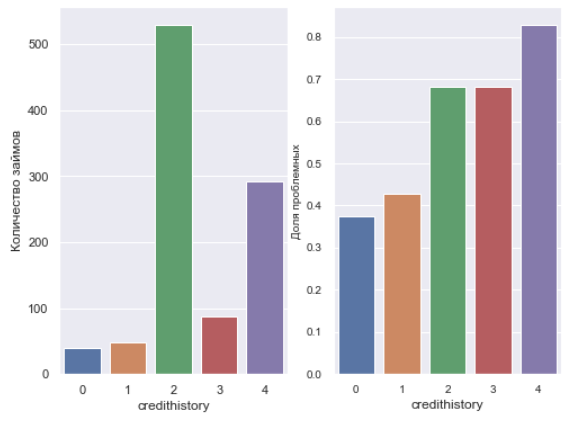


Рисунок Количество кредитов в зависимости от кредитной истории

Видно, что самая большая доля проблемных кредитов у заёмщиков 4 – у которых текущий счет с просрочкой или есть другие кредиты (не в этом банке).

Почти одинаковая доля проблемных кредитов у 2, 3 – т.е. у тех заёмщиков, у которых существующие кредиты возвращены должным образом на данный момент и у тех, у кого имеется задержка погашения в прошлом.

Выведем статистику по количеству людей, за которых отвечает заёмщик (peopleliable):

plt\_stat('peopleliable')

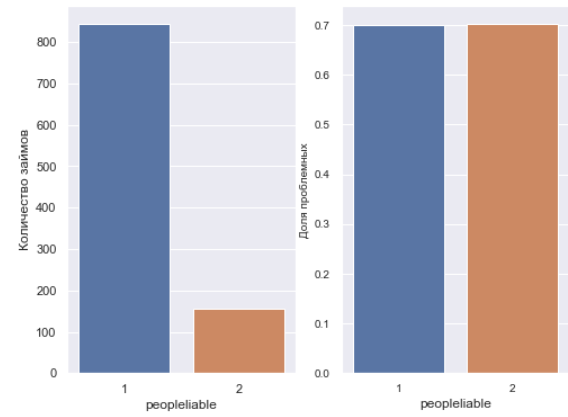


Рисунок Количество кредитов в зависимости от количества людей, за которых отвечает заёмщик

Доля невозврата кредита примерно одинакова в случае с 1, 2 людьми, хотя с двумя людьми количество заёмщиков в общем выборки намного меньше.

# Задача классификации

Зададим функции для оценки результатов построения – метрики accuracy, precision, recall и roc-auc.

# Функция для отчётов по оценке работы алгоритма

def get\_eval1(clf, x, y):

# Cross Validation to test and anticipate overfitting problem

scores1 = cross\_val\_score(clf, x, y, cv=2, scoring='accuracy')

scores2 = cross\_val\_score(clf, x, y, cv=2, scoring='precision')

scores3 = cross\_val\_score(clf, x, y, cv=2, scoring='recall')

scores4 = cross\_val\_score(clf, x, y, cv=2, scoring='roc\_auc')

# Средний скор и стандартное отклонение оценки скора

print("Cross Validation Accuracy: %0.2f (+/- %0.2f)" % (scores1.mean(), scores1.std()))

print("Cross Validation Precision: %0.2f (+/- %0.2f)" % (scores2.mean(), scores2.std()))

print("Cross Validation Recall: %0.2f (+/- %0.2f)" % (scores3.mean(), scores3.std()))

print("Cross Validation roc\_auc: %0.2f (+/- %0.2f)" % (scores4.mean(), scores4.std()))

return

def get\_eval2(clf, x\_train, y\_train, x\_test, y\_test):

# Cross Validation для тестирования и предотвращения переобучения

scores1 = cross\_val\_score(clf, x\_test, y\_test, cv=2, scoring='accuracy')

scores2 = cross\_val\_score(clf, x\_test, y\_test, cv=2, scoring='precision')

scores3 = cross\_val\_score(clf, x\_test, y\_test, cv=2, scoring='recall')

scores4 = cross\_val\_score(clf, x\_test, y\_test, cv=2, scoring='roc\_auc')

# Средний скор и стандартное отклонение оценки скора

print("Cross Validation Accuracy: %0.2f (+/- %0.2f)" % (scores1.mean(), scores1.std()))

print("Cross Validation Precision: %0.2f (+/- %0.2f)" % (scores2.mean(), scores2.std()))

print("Cross Validation Recall: %0.2f (+/- %0.2f)" % (scores3.mean(), scores3.std()))

print("Cross Validation roc\_auc: %0.2f (+/- %0.2f)" % (scores4.mean(), scores4.std()))

return

# Функция для получения roc-curve кривой

def get\_roc (y\_test,y\_pred):

# Compute ROC curve and ROC area for each class

fpr = dict()

tpr = dict()

roc\_auc = dict()

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred)

roc\_auc = auc(fpr, tpr)

#Plot of a ROC curve

plt.figure()

lw = 2

plt.plot(fpr, tpr, color='darkorange',

label='ROC curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.0])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="upper left")

plt.show()

return

Функция get\_eval1 считает метрики без разбиения на обучающую и тестовую выборки, а функция get\_eval2 – с разбиением.

Функция roc\_auc выводит кривую roc-auc графически.

Определим независимые переменные (матрица x) и зависимые переменные (матрица y). Разделим всю выборку на обучающую (train) и тестовую (train) для проведения кросс-валидации.

# Разделение датасета на подвыборки - обучающую и тестовую

# Unscaled, unnormalized data

x\_clean = data\_new.drop('classification', axis=1)

y\_clean = data\_new['classification']

x\_train\_clean, x\_test\_clean, y\_train\_clean, y\_test\_clean = train\_test\_split(x\_clean,y\_clean,test\_size=0.2, random\_state=1)

## Логистическая регрессия

Подключаем необходимые библиотеки для построения модели по алгоритму логистической регрессии, обучаем модель, а также выводим результаты работы функций get\_eval1, get\_eval2 и roc\_auc, а также коэффициенты регрессии.

Исходный код:

from sklearn.linear\_model import LogisticRegression

def logregclf(x\_train, y\_train, x\_test, y\_test):

print("LogisticRegression")

model = LogisticRegression().fit(x\_train, y\_train)

print(model,'\n')

y\_pred = model.predict\_proba(x\_test)[:,1]

y\_p = model.predict(x\_test)

get\_eval1(model, x\_clean, y\_clean)

get\_eval2(model, x\_train, y\_train, x\_test, y\_test)

get\_roc (y\_test,y\_pred)

print(cr(y\_p, y\_test))

np.set\_printoptions(suppress=True)

print('Коэффициенты: ', model.coef\_, ', свободный член: ', model.intercept\_)

return

logregclf(x\_train\_clean, y\_train\_clean, x\_test\_clean, y\_test\_clean)

Результат выполнения:

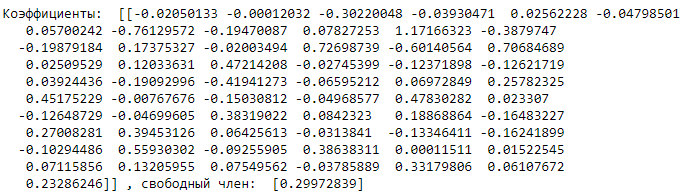


Рисунок Коэффициенты логистической регрессии

Видно, какие переменные коррелируют положительно, а какие отрицательно. Поскольку переменные нормализованы, то признаки с корреляцией 1.17, 0.72 влияют намного больше на модель, чем признак с корреляцией 0.0001.

## Нейронные сети

Основу нейронной сети представляет искусственный нейрон или просто нейрон. На основе нейронов формируются персептроны. Они состоят из одного слоя искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов.

Персептрон классифицирует значения входных сигналов (входов), принадлежащих n-мерному гиперпространству по некоторому числу классов. Т.е. нейронная сеть или персептрон - это алгоритм, использующий уравнение линейного неравенства (линейного фильтра), с помощью которого можно причислить исследуемый объект к тому или иному классу.

В пакете sklearn в модуле нейронных сетей (neural\_network) содержатся методы, основанные на нейронных сетях, в частности, многослойный персептрон (Multi-layer Perceptron classifier).

В качестве функций активации будем использовать логистическую функцию



А также гиперболический тангенс (tanh)

Сравним оценку кросс-валидации для различных функций активации.

1. Логистическая функция активации

Подключаем необходимые библиотеки для построения модели, обучаем модель, далее выводим результаты работы функций get\_eval1, get\_eval2 и roc\_auc

Исходный код:

from sklearn.neural\_network import MLPClassifier

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

print("MLPClassifier")

clf = MLPClassifier(activation='logistic', solver='adam', random\_state=1, max\_iter=300).fit(x\_train\_clean, y\_train\_clean)

clf.predict\_proba(x\_test\_clean[:1])

clf.predict(x\_test\_clean[:5])

clf.score(x\_test\_clean, y\_test\_clean)

model=clf

y\_pred = model.predict\_proba(x\_test\_clean)[:,1]

y\_p = model.predict(x\_test\_clean)

get\_eval1(model, x\_clean, y\_clean)

get\_eval2(model, x\_train\_clean, y\_train\_clean, x\_test\_clean, y\_test\_clean)

get\_roc (y\_test\_clean,y\_pred)

print(cr(y\_p, y\_test\_clean))

np.set\_printoptions(suppress=True)

from sklearn.neural\_network import MLPClassifier

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

1. Функция активации tanh

Подключаем необходимые библиотеки для построения модели, обучаем модель, далее выводим результаты работы функций get\_eval1, get\_eval2 и roc\_auc

Исходный код:

print("MLPClassifier")

clf = MLPClassifier(activation='tanh', random\_state=1, max\_iter=300).fit(x\_train\_clean, y\_train\_clean)

clf.predict\_proba(x\_test\_clean[:1])

clf.predict(x\_test\_clean[:5])

clf.score(x\_test\_clean, y\_test\_clean)

model=clf

y\_pred = model.predict\_proba(x\_test\_clean)[:,1]

y\_p = model.predict(x\_test\_clean)

get\_eval1(model, x\_clean, y\_clean)

get\_eval2(model, x\_train\_clean, y\_train\_clean, x\_test\_clean, y\_test\_clean)

get\_roc (y\_test\_clean,y\_pred)

print(cr(y\_p, y\_test\_clean))

np.set\_printoptions(suppress=True)

Оценка эффективности моделей

Введём концепцию для описания метрик в терминах ошибок классификации — confusion matrix (матрица ошибок).

Матрица ошибок классификации выглядит следующим образом:

Таблица 1 Матрица ошибок классификации

|  |  |  |
| --- | --- | --- |
|  | *α(x) =* 1 | *α(x) =* 0 |
| *y* = 1 | TP | FN |
| *y* = 0 | FP | TN |

(TP – сокращение для True Positive, FN – False Negative, и аналогично в двух оставшихся случаях).

Описание основных метрик:

показывает долю правильных ответов, т.е. долю правильно классифицированных заёмщиков.

точность - доля объектов, названных классификатором положительными и при этом действительно являющимися положительными или доля верно классифицированных дефолтов среди таких заёмщиков, которых алгоритм посчитал дефолтными.

полнота, показывающая долю верно указанных дефолтов.

доля хороших кредитов, неправильно отнесенных к плохим.

Среднее гармоническое точности и полноты, показывает сбалансированность алгоритма, являясь обобщением точности и полноты.

Кривая ошибок (Receiver Operating Characteristic, ROC) отображает изменение отношения доли Recall верно классифицированных плохих кредитов в их общем количестве к доле False Positive Rate хороших кредитов, неправильно отнесенных к плохим, при варьировании порога решающего правила.

Этот график, позволяет оценить качество бинарной классификации.

Площадь под кривой ошибок AUC служит показателем качества алгоритма. Равенство AUC = 0,5 означает, что алгоритм относит объекты к категориям наугад. Чем больше AUC, тем качественнее алгоритм.

Визуализируем кривые ROC-AUC:

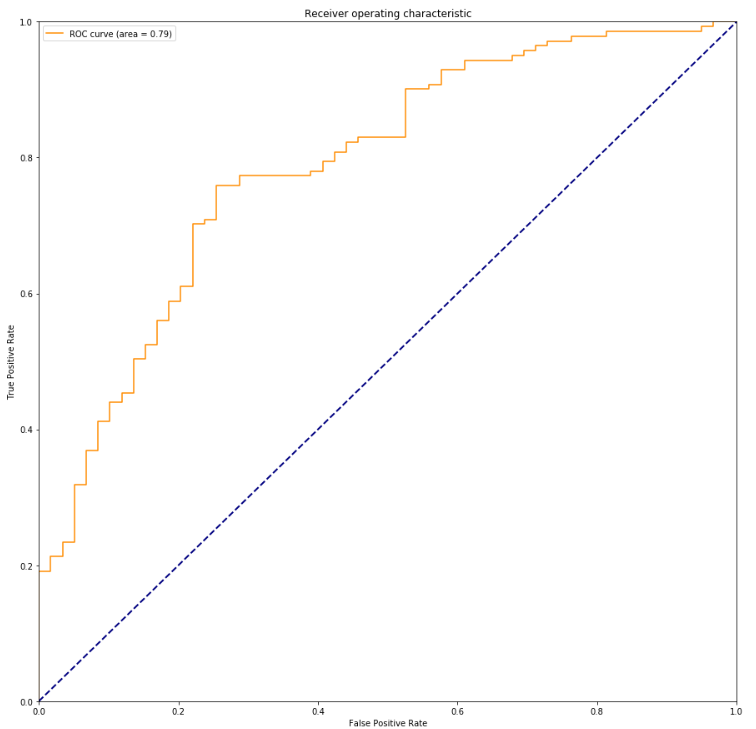


Рисунок ROC-AUC LogisticRegression

|  |  |
| --- | --- |
| Рисунок 13 ROC-AUC MLPClassifier activation='logistic' | Рисунок 15 ROC-AUC MLPClassifier activation='tanh' |

По ROC-AUC кривой видно, что выбор функции активации влияет на кривую ошибок (рис. 13, 14).

ROC-AUC кривые достаточно похожи у логистической регрессии и у многослойного персептрона с функцией активации tanh.

Сводная информация по разным методикам оценивания:

Таблица 2 Оценка эффективности моделей

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Metrics | Logistic Regression | | MLPClassifier  activation='logistic' | | MLPClassifier  activation='tanh' | |
| Score | STD | Score | STD | Score | STD |
| Cross Validation Accuracy | 0.74 | 0.01 | 0.72 | 0.02 | 0.70 | 0 |
| Cross Validation Precision | 0.78 | 0.01 | 0.73 | 0.03 | 0.70 | 0 |
| Cross Validation Recall | 0.87 | 0 | 0.98 | 0 | 1 | 0 |
| Cross Validation roc\_auc | 0.77 | 0 | 0.76 | 0.10 | 0.66 | 0.04 |
| f-1 score | 0.78 | | 0.81 | | 0.83 | |

Из табл. 2 видно, что на результат классификации в случае нейронных сетей (MLPClassifier) повлиял выбор функции активации – с гиперболическим тангенсом результат получился лучше, recall оценка = 1, т.е. 100% дефолтов указаны верно.

Таким образом, можно сказать, что нейронные сети (многослойный персептрон) показали себя лучше логистической регрессии.

# Список использованных источников

1. German Credit Data [Электронный ресурс]// https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)
2. Электронный учебник по статистике Statsoft - 2М гистограммы [Электронный ресурс]// <http://statsoft.ru/home/textbook/default.htm>
3. Кондрашов, Ю.Н. Анализ данных и машинное обучение на платформе MS SQL Server: учебное пособие / Кондрашов Ю.Н. — Москва: Русайнс, 2019. — 303 с. — ISBN 978-5-4365-3369-8
4. Cramer, J. S. (2002). The origins of logistic regression (Technical report). 119. Tinbergen Institute. pp. 167–178. doi:10.2139/ssrn.360300 [Электронный ресурс]// <https://papers.tinbergen.nl/02119.pdf>
5. Богданов Л.Ю. Оценка эффективности бинарных классификаторов на основе логистической регрессии методом roc-анализа// Вестник Саратовского государственного технического университета. 2010. №40. [Электронный ресурс]// <https://cyberleninka.ru/article/n/otsenka-effektivnosti-binarnyh-klassifikatorov-na-osnove-logisticheskoy-regressii-metodom-roc-analiza>
6. Sklearn – краткий гид пользователя [Электронный ресурс]// <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.neural_network>
7. Метрики в машинном обучении [Электронный ресурс]// <https://habr.com/ru/company/ods/blog/328372>
8. Zweig, Mark H.; Campbell, Gregory. Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine (англ.) // Clinical Chemistry : journal. — 1993. — Vol. 39, no. 8. — P. 561—577. — PMID 8472349