Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс   
«Технологии машинного обучения»  
 Отчет по лабораторной работе №5

«Разведочный анализ данных. Исследование и визуализация данных.»

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-63Б | преподаватель каф. ИУ5 |
| Тарновский Д.Р. | Гапанюк Ю.Е. |

Москва, 2022 г.

In [3]:

*#Датасет содержит данные о кредитах на покупку электроники, которые были*

**import** pandas **as** pd

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split, GridSearchCV, Rand

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.preprocessing **import** MinMaxScaler, StandardScaler

**from** sklearn.linear\_model **import** LogisticRegression, LogisticRegressionC **from** sklearn.ensemble **import** RandomForestClassifier, GradientBoostingCla **from** sklearn.metrics **import** accuracy\_score, precision\_score, recall\_scor **from** sklearn.neural\_network **import** MLPClassifier

**from** warnings **import** simplefilter

simplefilter('ignore')

In [4]:

*# записываем CSV-файл в объект DataFrame*

data **=** pd**.**read\_csv('credit\_train\_preprocess.csv', encoding**=**'cp1251', sep

In [5]:

*# смотрим на первые пять строк*

data**.**head() data**.**info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 170746 entries, 0 to 170745 Data columns (total 39 columns):

# Column Non-Null Count Dtype

|  |  |  |  |
| --- | --- | --- | --- |
| 0 age | 170746 | non-null | float64 |
| 1 credit\_sum | 170746 | non-null | float64 |
| 2 credit\_month | 170746 | non-null | int64 |
| 3 tariff\_id | 170746 | non-null | float64 |
| 4 score\_shk | 170746 | non-null | float64 |
| 5 monthly\_income | 170746 | non-null | float64 |
| 6 credit\_count | 170746 | non-null | float64 |
| 7 overdue\_credit\_count | 170746 | non-null | float64 |
| 8 open\_account\_flg | 170746 | non-null | int64 |
| 9 gender\_F | 170746 | non-null | int64 |
| 10 gender\_M | 170746 | non-null | int64 |
| 11 job\_position\_ATP | 170746 | non-null | int64 |
| 12 job\_position\_BIS | 170746 | non-null | int64 |
| 13 job\_position\_BIU | 170746 | non-null | int64 |
| 14 job\_position\_DIR | 170746 | non-null | int64 |
| 15 job\_position\_HSK | 170746 | non-null | int64 |
| 16 job\_position\_INP | 170746 | non-null | int64 |
| 17 job\_position\_INV | 170746 | non-null | int64 |
| 18 job\_position\_NOR | 170746 | non-null | int64 |
| 19 job\_position\_ONB | 170746 | non-null | int64 |
| 20 job\_position\_PNA | 170746 | non-null | int64 |
| 21 job\_position\_PNI | 170746 | non-null | int64 |
| 22 job\_position\_PNS | 170746 | non-null | int64 |
| 23 job\_position\_PNV | 170746 | non-null | int64 |

|  |  |  |  |
| --- | --- | --- | --- |
| 24 job\_position\_SPC | 170746 | non-null | int64 |
| 25 job\_position\_UMN | 170746 | non-null | int64 |
| 26 job\_position\_WOI | 170746 | non-null | int64 |
| 27 job\_position\_WRK | 170746 | non-null | int64 |
| 28 job\_position\_WRP | 170746 | non-null | int64 |
| 29 education\_ACD | 170746 | non-null | int64 |
| 30 education\_GRD | 170746 | non-null | int64 |
| 31 education\_PGR | 170746 | non-null | int64 |
| 32 education\_SCH | 170746 | non-null | int64 |
| 33 education\_UGR | 170746 | non-null | int64 |
| 34 marital\_status\_CIV | 170746 | non-null | int64 |
| 35 marital\_status\_DIV | 170746 | non-null | int64 |
| 36 marital\_status\_MAR | 170746 | non-null | int64 |
| 37 marital\_status\_UNM | 170746 | non-null | int64 |
| 38 marital\_status\_WID | 170746 | non-null | int64 |

In [7]:

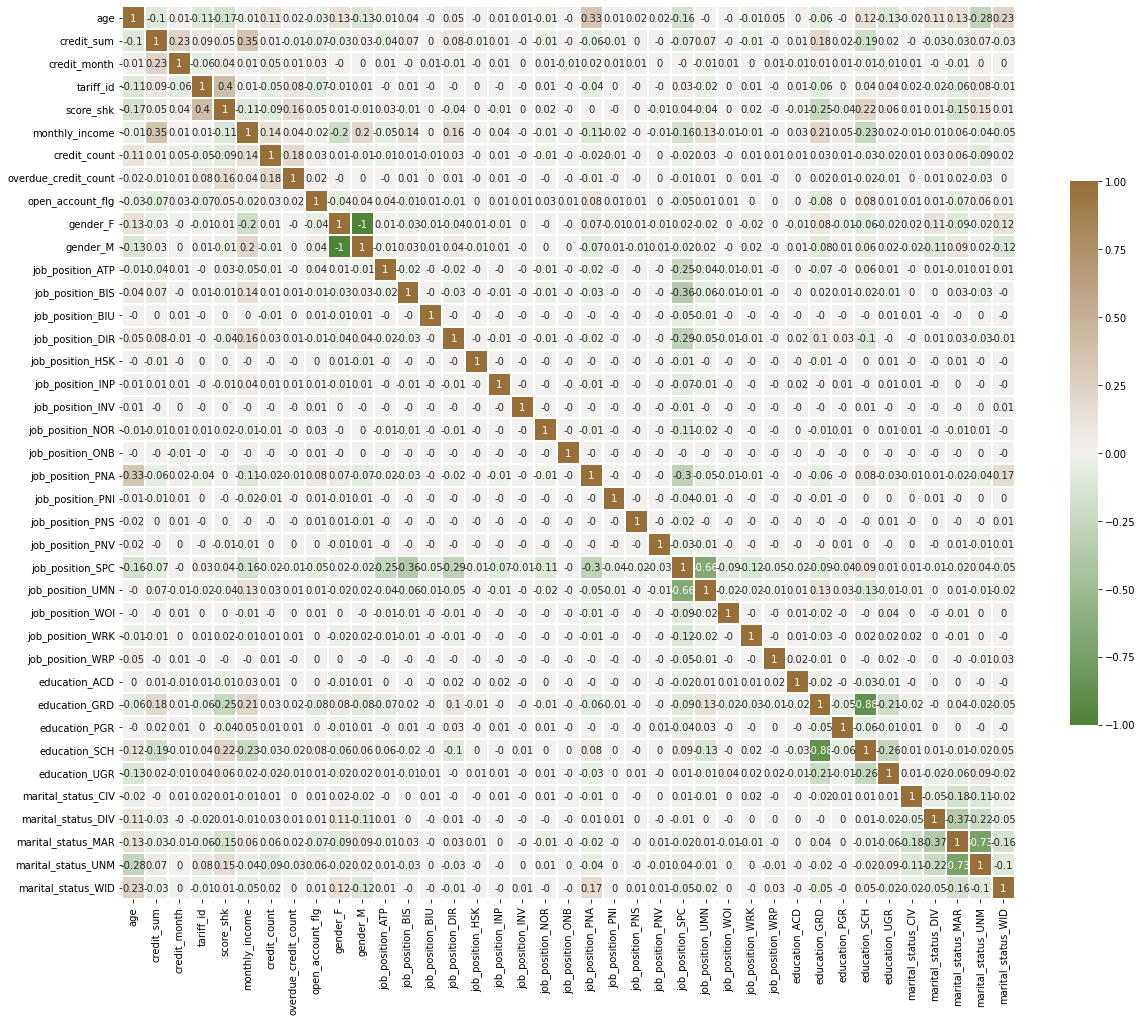
corr **=** data**.**corr()**.**round(2)

f, ax **=** plt**.**subplots(figsize**=**(20, 20))

cmap **=** sns**.**diverging\_palette(120, 50, as\_cmap**=True**) sns**.**heatmap(data**=**corr, cmap**=**cmap, annot**=True**, vmax**=**1.0, square**=True**, lin plt**.**show()

dtypes: float64(7), int64(32) memory usage: 50.8 MB

# Корреляционный анализ



In [8]:

print('Признаки, имеющие максимальную по модулю корреляцию с целевым при best\_params **=** data**.**corr()['open\_account\_flg']**.**map(abs)**.**sort\_values(ascen

best\_params **=** best\_params[best\_params**.**values **>** 0.02] best\_params

Out[8]:

In [9]:

Признаки, имеющие максимальную по модулю корреляцию с целевым признаком education\_GRD 0.082371

education\_SCH 0.078337

job\_position\_PNA 0.076889

credit\_sum 0.072039

tariff\_id 0.067346

marital\_status\_MAR 0.067112

marital\_status\_UNM 0.061312

job\_position\_SPC 0.049143

score\_shk 0.048686

gender\_F 0.044265

gender\_M 0.044265

job\_position\_ATP 0.038288

credit\_count 0.032374

age 0.031062

job\_position\_NOR 0.027320

credit\_month 0.025809

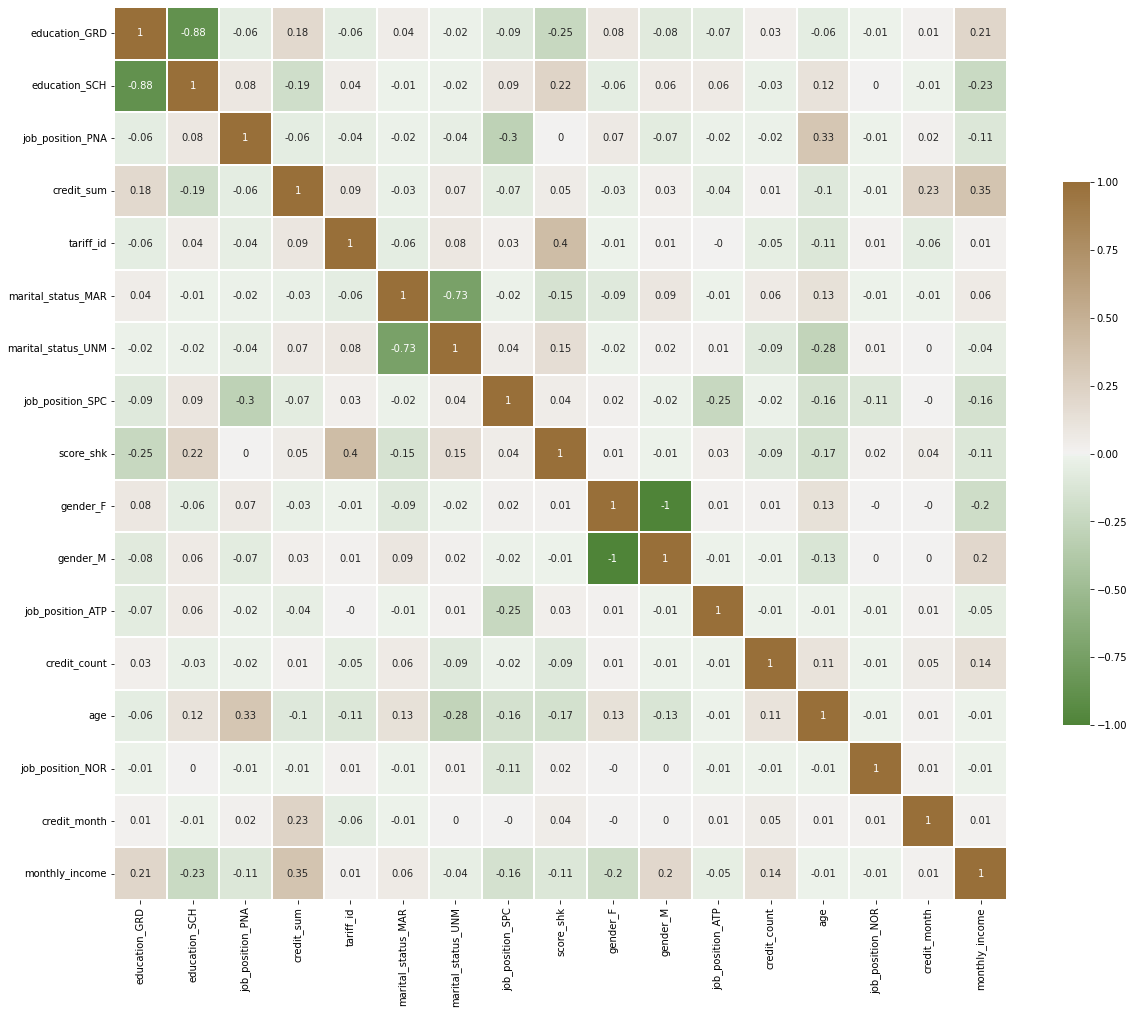
monthly\_income 0.023697

Name: open\_account\_flg, dtype: float64

corr **=** data[best\_params**.**index]**.**corr()**.**round(2) f, ax **=** plt**.**subplots(figsize**=**(20, 20))

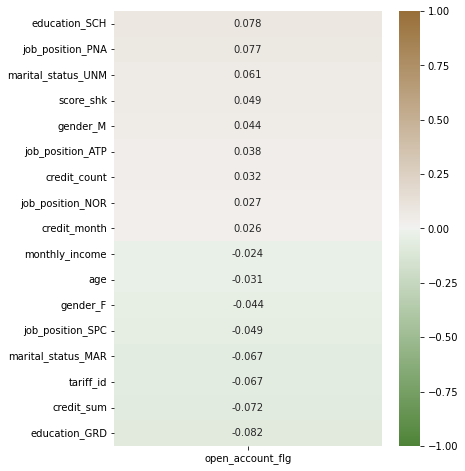
cmap **=** sns**.**diverging\_palette(120, 50, as\_cmap**=True**)

sns**.**heatmap(data**=**corr, cmap**=**cmap, annot**=True**, vmax**=**1.0, square**=True**, lin plt**.**show()



In [10]:

plt**.**figure(figsize**=**(6, 8)) sns**.**heatmap(pd**.**DataFrame(data[np**.**append(best\_params**.**index**.**values, 'open\_ plt**.**show()



In [11]:

data\_best **=** data[best\_params**.**index] data\_best**.**head()

# Разделение выборки на обучающую и тестовую

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Out[11]: | **education\_GRD** | **education\_SCH** | **job\_position\_PNA** | **credit\_sum** | **tariff\_id** | **marital\_status\_M** |
|  | **0** 1 | 0 | 0 | 59998.00 | 1.6 |  |
|  | **1** 0 | 1 | 0 | 10889.00 | 1.1 |  |
|  | **2** 0 | 1 | 0 | 10728.00 | 1.1 |  |
|  | **3** 0 | 1 | 0 | 12009.09 | 1.1 |  |
|  | **4** 0 | 1 | 0 | 21229.00 | 1.1 |  |

In [12]:

y **=** data['open\_account\_flg']

*#X = data.drop('open\_account\_flg', axis=1)*

X **=** data\_best

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.75 x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x\_train, y\_train, te

# Масштабирование данных

In [13]:

scaler **=** MinMaxScaler()**.**fit(x\_train)

x\_train **=** pd**.**DataFrame(scaler**.**transform(x\_train), columns**=**x\_train**.**column x\_test **=** pd**.**DataFrame(scaler**.**transform(x\_test), columns**=**x\_train**.**columns) x\_train**.**describe()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Out[13]: | **education\_GRD** | **education\_SCH** | **job\_position\_PNA** | **credit\_sum** | **tariff\_id** | **mari** |
|  | **count** 29880.000000 | 29880.000000 | 29880.000000 | 29880.000000 | 29880.000000 |  |
|  | **mean** 0.425000 | 0.514759 | 0.023561 | 0.117340 | 0.345539 |  |
|  | **std** 0.494351 | 0.499790 | 0.151679 | 0.082275 | 0.252486 |  |
|  | **min** 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |  |
|  | **25%** 0.000000 | 0.000000 | 0.000000 | 0.060249 | 0.106383 |  |
|  | **50%** 0.000000 | 1.000000 | 0.000000 | 0.092536 | 0.340426 |  |
|  | **75%** 1.000000 | 1.000000 | 0.000000 | 0.148270 | 0.638298 |  |
|  | **max** 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |  |

In [82]:

**from** sklearn.metrics **import** mean\_absolute\_error

**from** sklearn.metrics **import** median\_absolute\_error, r2\_score

**def** print\_metrics(y\_test, y\_pred):

print(f"Precision: {precision\_score(y\_test, y\_pred)}") print(f"F1-measure: {f1\_score(y\_test, y\_pred)}")

print('mean\_absolute\_error: {}'**.**format(round(mean\_absolute\_error(y\_t print('median\_absolute\_error: {}'**.**format(round(median\_absolute\_error print('r2\_score: {}'**.**format(round(r2\_score(y\_test, y\_pred), 2)))

# Модель №1: Случайный лес

In [15]:

print\_metrics(y\_test, RandomForestClassifier(random\_state**=**17)**.**fit(x\_trai

In [90]:

Precision: 0.4737991266375546

F1-measure: 0.15617128463476068

# Подбор гиперпараметров

{'max\_features': 'auto', 'min\_samples\_leaf': 5, 'n\_estimators': 1000}

rf **=** RandomForestClassifier(random\_state**=**17) params **=** {'n\_estimators': [100, 1000],

'max\_features': ['auto', 'sqrt'], 'min\_samples\_leaf': [1, 3, 5 grid\_cv **=** GridSearchCV(estimator**=**rf, cv**=**5, param\_grid**=**params, n\_jobs**=-**1, grid\_cv**.**fit(x\_train, y\_train)

print(grid\_cv**.**best\_params\_)

In [ ]:

best\_rf **=** grid\_cv**.**best\_estimator\_ best\_rf**.**fit(x\_train, y\_train) y\_pred\_rf **=** best\_rf**.**predict(x\_test) print\_metrics(y\_test, y\_pred\_rf)

In [16]:

print\_metrics(y\_test, GradientBoostingClassifier(random\_state**=**17)**.**fit(x\_

# Модель №2: Градиентный бустинг

In [76]:

Accuracy: 0.8218803685772295

Precision: 0.5819672131147541

Recall: 0.06118052563550194

F1-measure: 0.11072124756335285

# Подбор гиперпараметров

{'min\_samples\_leaf': 5, 'n\_estimators': 200}

gb **=** GradientBoostingClassifier(random\_state**=**17)

params **=** {'n\_estimators': [10, 50, 100, 200], 'min\_samples\_leaf': [1, 3, grid\_cv **=** GridSearchCV(estimator**=**gb, cv**=**5, param\_grid**=**params, n\_jobs**=-**1, grid\_cv**.**fit(x\_train, y\_train)

print(grid\_cv**.**best\_params\_)

In [77]:

best\_gb **=** grid\_cv**.**best\_estimator\_ best\_gb**.**fit(x\_train, y\_train) y\_pred\_gb **=** best\_gb**.**predict(x\_test) print\_metrics(y\_test, y\_pred\_gb)

In [24]:

dataset **=** Dataset(x\_train, y\_train, x\_test)

Precision: 0.5709876543209876

F1-measure: 0.13988657844990549

# Модель №3: Стекинг

In [97]:

**from** sklearn.ensemble **import** RandomForestClassifier, StackingClassifier,

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.linear\_model **import** SGDClassifier **from** sklearn.neighbors **import** KNeighborsClassifier **from** sklearn.tree **import** DecisionTreeClassifier

layer\_one\_estimators **=** [

('rf\_1', RandomForestClassifier(n\_estimators**=**10, ('rf\_3', GradientBoostingClassifier(n\_estimators

]

layer\_two\_estimators **=** [

('dt\_2', DecisionTreeClassifier()),

('rf\_2', RandomForestClassifier(n\_estimators**=**10, ('rf\_4', GradientBoostingClassifier(n\_estimators

]

layer\_two **=** StackingClassifier(estimators**=**layer\_two\_estimators, final\_es

*# Create Final model by*

clf **=** StackingClassifier(estimators**=**layer\_one\_estimators, final\_estimato

*#layer\_2 = StackingClassifier(estimators=profi\_learners, final\_estimator #layer\_1 = StackingClassifier(estimators=base\_learners, final\_estimator=*

clf**.**fit(x\_train, y\_train)

Out[97]:

StackingClassifier(estimators=[('rf\_1',

RandomForestClassifier(n\_estimators=10,

random\_state=42)),

00))],

2',

('rf\_3',

GradientBoostingClassifier(n\_estimators=2 final\_estimator=StackingClassifier(estimators=[('dt\_

Decisi

onTreeClassifier()), 2',

ForestClassifier(n\_estimators=10,

('rf\_ Random

In [98]:

random\_state=42)), 4',

ntBoostingClassifier(n\_estimators=20))],

Classifier(random\_state=1488)))

print\_metrics(y\_test, clf**.**predict(x\_test))

('rf\_ Gradie

final\_estimator=MLP

In [6]:

Precision: 0.6148148148148148

F1-measure: 0.12813585488228482

mean\_absolute\_error: 0.18

median\_absolute\_error: 0.0

r2\_score: -0.19

# Сравнение моделей

Случайный лес

print("Случайный лес") print\_metrics(y\_test, y\_pred\_rf)

print("\nГрадиентный бустинг") print\_metrics(y\_test, y\_pred\_gb)

print("\nСтекинг") print\_metrics(y\_test, y\_pred\_stack)

**~\AppData\Local\Temp/ipykernel\_12728/287930607.py** in <module>

1 print**("Случайный лес")**

**----> 2** print\_metrics**(**y\_test**,** y\_pred\_rf**)** 3

1. print**("\nГрадиентный бустинг")**
2. print\_metrics**(**y\_test**,** y\_pred\_gb**)**

**NameError**: name 'print\_metrics' is not defined

Traceback (most recent call las

**--**

**NameError**

t)

In [ ]: