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 №4

«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 [21]:

*#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.metrics **import** accuracy\_score, precision\_score, recall\_scor **from** sklearn.tree **import** DecisionTreeClassifier, export\_graphviz, export **from** sklearn.svm **import** SVC

**import** graphviz

**from** IPython.display **import** Image **from** warnings **import** simplefilter simplefilter('ignore')

In [2]:

*# записываем CSV-файл в объект DataFrame*

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

In [3]:

*# смотрим на первые пять строк*

data**.**head()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[3]: | **age** | **credit\_sum** | **credit\_month** | **tariff\_id** | **score\_shk** | **monthly\_income** | **credit\_count** | **over** |
|  | **0** 34.0 | 59998.00 | 10 | 1.6 | 0.461599 | 30000.0 | 1.0 |  |
|  | **1** 34.0 | 10889.00 | 6 | 1.1 | 0.461599 | 35000.0 | 2.0 |  |
|  | **2** 32.0 | 10728.00 | 12 | 1.1 | 0.461599 | 35000.0 | 5.0 |  |
|  | **3** 27.0 | 12009.09 | 12 | 1.1 | 0.461599 | 35000.0 | 2.0 |  |
|  | **4** 45.0 | 21229.00 | 10 | 1.1 | 0.421385 | 35000.0 | 1.0 |  |

5 rows × 39 columns

In [4]:

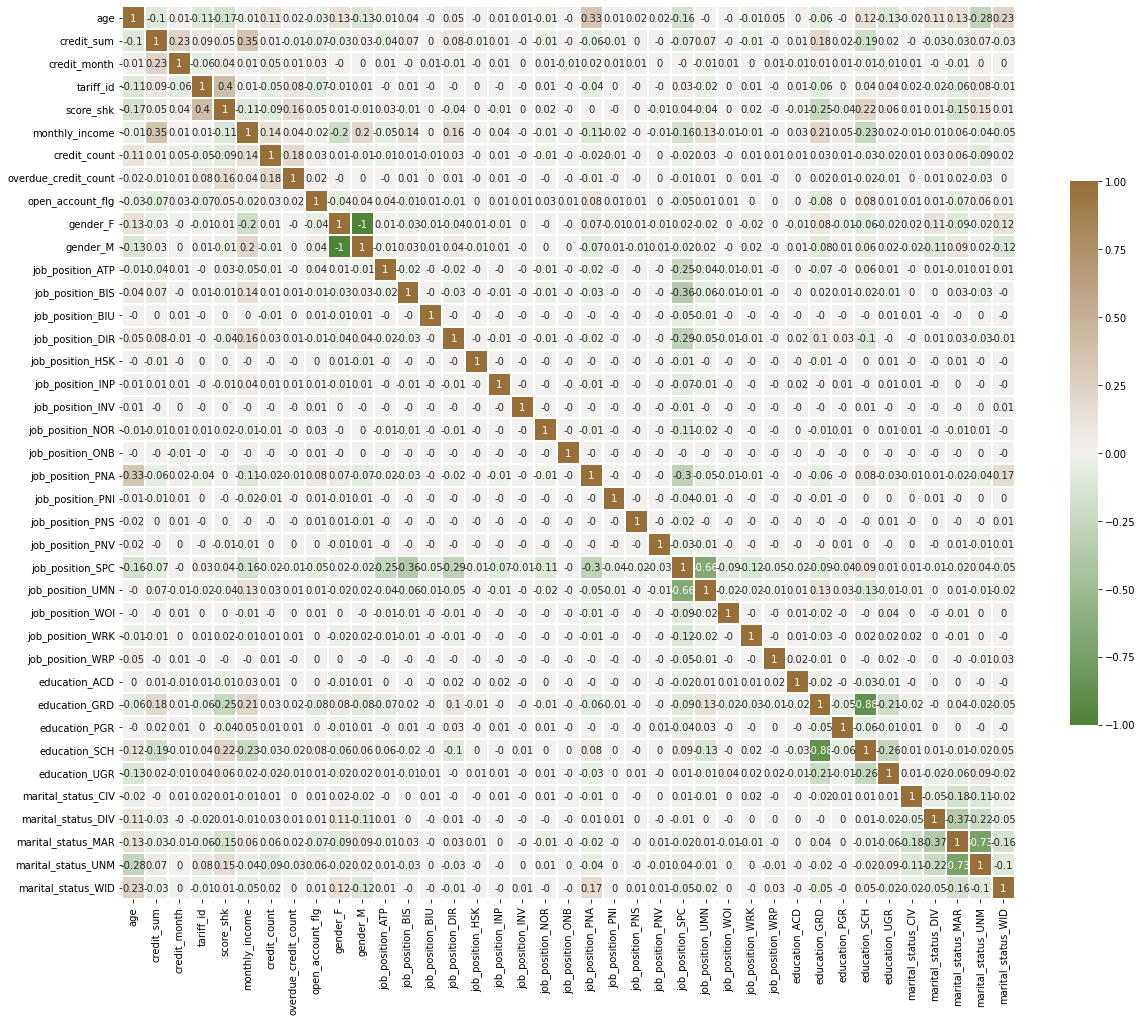
corr **=** data**.**corr()**.**round(2)

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()

# Корреляционный анализ



In [5]:

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[5]:

In [6]:

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

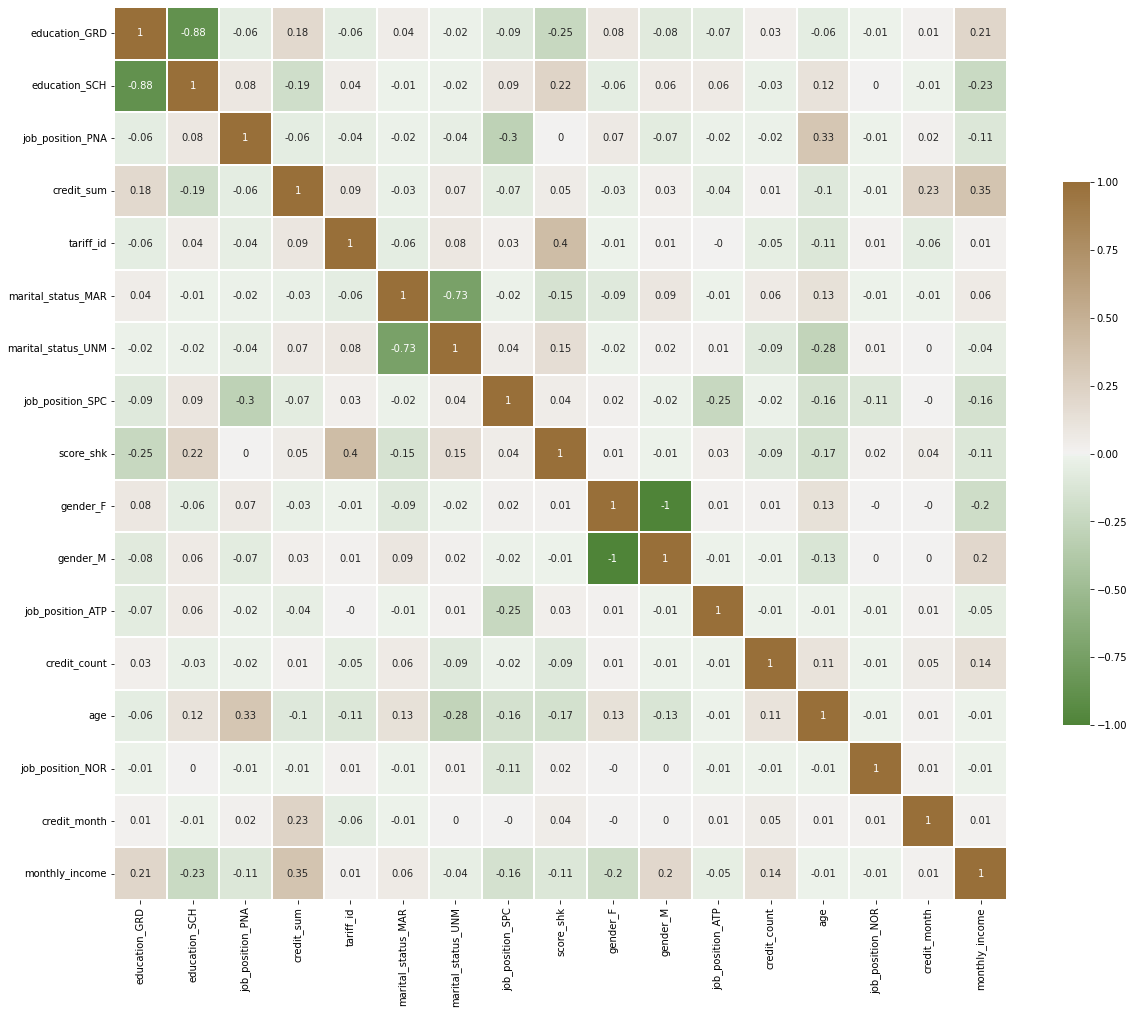
monthly\_income 0.023697

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**)

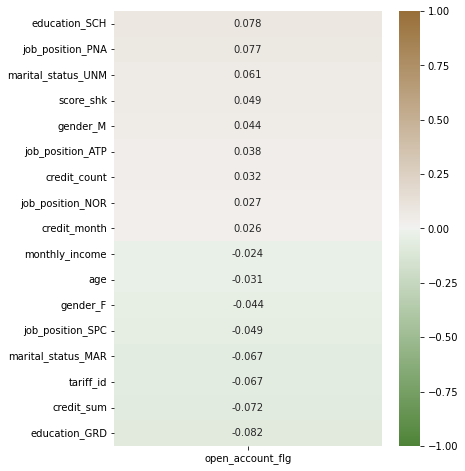
sns**.**heatmap(data**=**corr, cmap**=**cmap, annot**=True**, vmax**=**1.0, square**=True**, lin plt**.**show()



In [7]:

plt**.**figure(figsize**=**(6, 8))

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



In [8]:

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

# Разделение выборки на обучающую и тестовую

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out[8]: | **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 [9]:

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

In [10]:

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

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

# Масштабирование данных

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Out[10]: | **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 [11]:

*# обучаем логистическую регрессию*

clf **=** LogisticRegression() clf**.**fit(x\_train, y\_train);

# Логистическая регрессия

In [12]:

*#считаем точность (долю правильных ответов) на тестовой выборке*

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

Out[12]:

In [13]:

**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)}")

0.8191472747149774

In [14]:

*# вычисляем метрики точности*

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

In [15]:

Precision: 0.5862068965517241

F1-measure: 0.01446808510638298

# SVC

Precision: 0.5714285714285714

best\_svm\_model **=** SVC()

best\_svm\_model**.**fit(x\_train, y\_train)

y\_pred\_svm **=** best\_svm\_model**.**predict(x\_test) print\_metrics(y\_test, y\_pred\_svm)

F1-measure: 0.003436426116838488

# Дерево решений

In [16]:

params **=** {'min\_samples\_leaf': range(3, 30)}

tree **=** DecisionTreeClassifier(random\_state**=**3) *#max\_depth=5*

grid\_cv **=** GridSearchCV(estimator**=**tree, cv**=**5, param\_grid**=**params, n\_jobs**=-** grid\_cv**.**fit(x\_train, y\_train)

*#tree.fit(x\_train, y\_train)*

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

{'min\_samples\_leaf': 29}

In [17]:

best\_tree **=** grid\_cv**.**best\_estimator\_ best\_tree**.**fit(x\_train, y\_train)

y\_pred\_tree **=** best\_tree**.**predict(x\_test) print\_metrics(y\_test, y\_pred\_tree)

In [18]:

Precision: 0.4374255065554231

F1-measure: 0.23227848101265824

importances **=** pd**.**DataFrame(data**=**zip(x\_train**.**columns, best\_tree**.**feature\_i print('Важность признаков в дереве решений\n')

**for** row **in** importances**.**sort\_values(by**=**'Важность', ascending**=False**)**.**value

print(f'{row[0]}: {round(row[1], 3)}')

Важность признаков в дереве решений

In [19]:

tariff\_id: 0.203

credit\_sum: 0.191

score\_shk: 0.169

age: 0.11

credit\_month: 0.102

monthly\_income: 0.058

credit\_count: 0.045

job\_position\_PNA: 0.03

education\_SCH: 0.027

marital\_status\_MAR: 0.017

gender\_F: 0.012

gender\_M: 0.012

job\_position\_SPC: 0.01

marital\_status\_UNM: 0.009

education\_GRD: 0.005

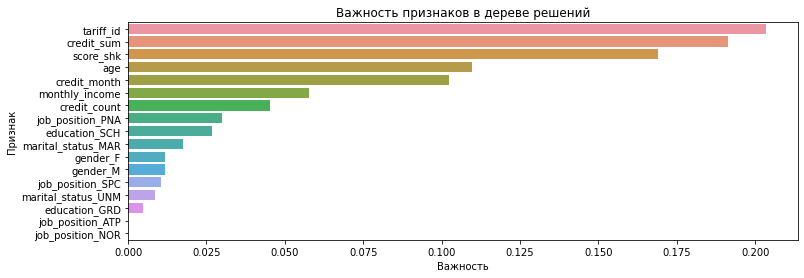
job\_position\_ATP: 0.0

job\_position\_NOR: 0.0

plt**.**figure(figsize**=**(12, 4))

sns**.**barplot(data**=**importances**.**sort\_values(by**=**'Важность', ascending**=False**) plt**.**title('Важность признаков в дереве решений')

plt**.**show()



In [31]:

**from** sklearn **import** tree

fig, ax **=** plt**.**subplots(figsize**=**(30, 15))

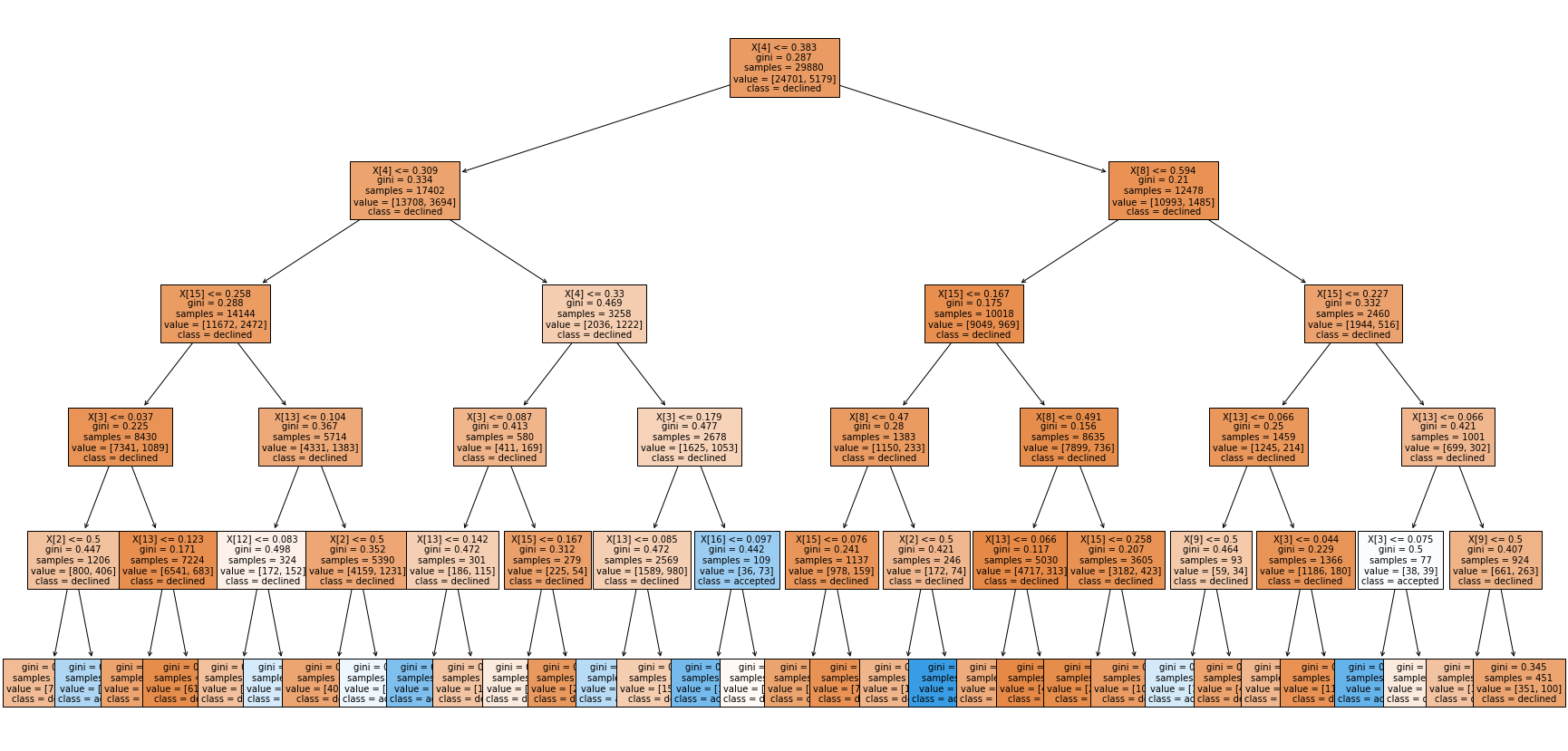
clf **=** DecisionTreeClassifier(max\_depth **=** 5,

random\_state **=** 0)

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

cn**=**['declined', 'accepted']

tree**.**plot\_tree(clf, fontsize**=**10, class\_names**=**cn, filled**=True**) plt**.**show()



In [32]:

# Сравнение моделей

Логистическая регрессия

print('Логистическая регрессия') print\_metrics(y\_test, y\_pred\_log)

print('\nМетод опорных векторов') print\_metrics(y\_test, y\_pred\_svm)

print('\nДерево решений')

print\_metrics(y\_test, y\_pred\_tree)

Precision: 0.5862068965517241

F1-measure: 0.01446808510638298

Метод опорных векторов

Precision: 0.5714285714285714

F1-measure: 0.003436426116838488

Дерево решений

Precision: 0.4374255065554231

F1-measure: 0.23227848101265824

In [ ]: