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()
```

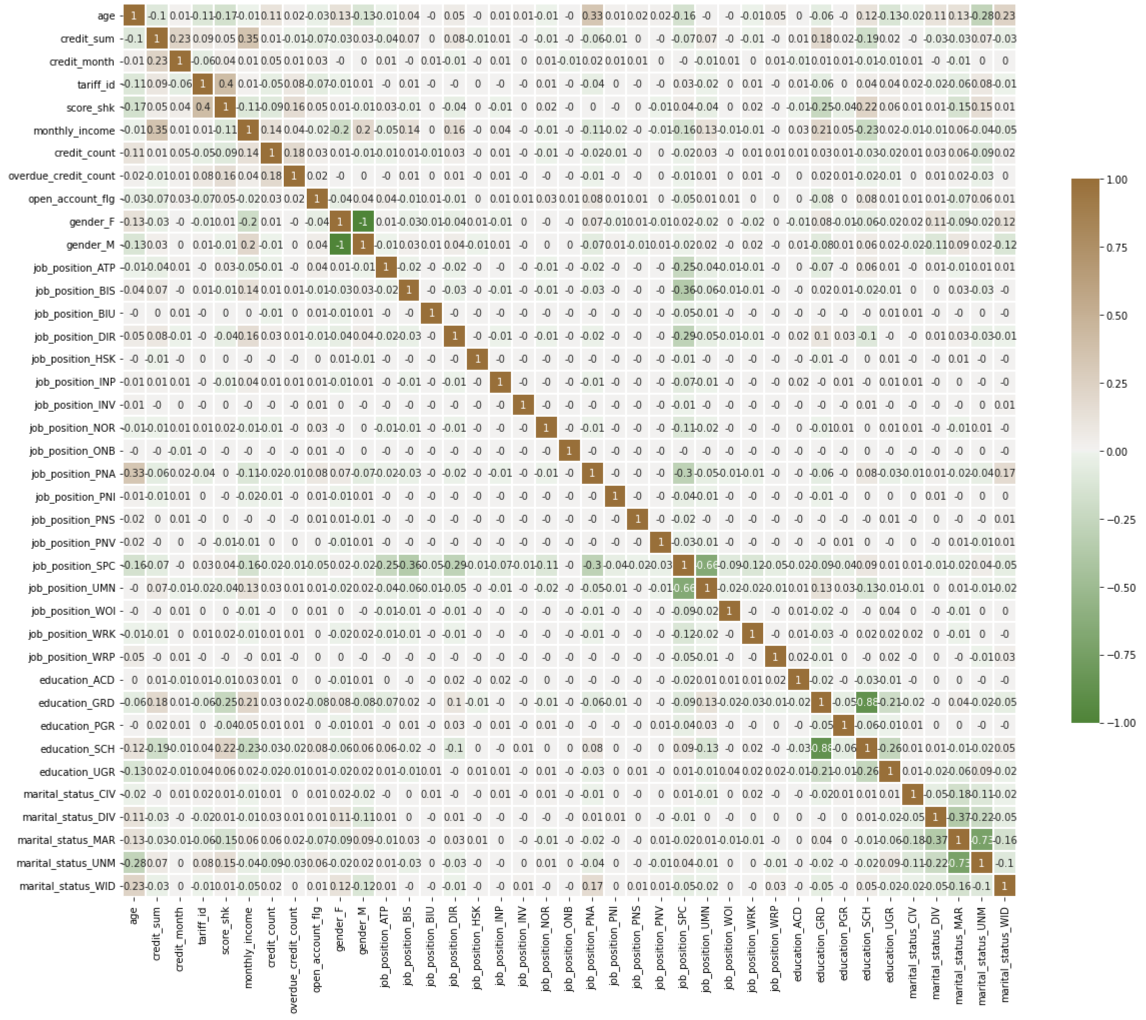
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170746 entries, 0 to 170745
Data columns (total 39 columns):
 #   Column                Non-Null Count   Dtype
---------------------------------------------------------
 0   age                   170746 non-null  float64
 1   credit_sum            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

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

1) Корреляционный анализ

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()
```



In [8]:

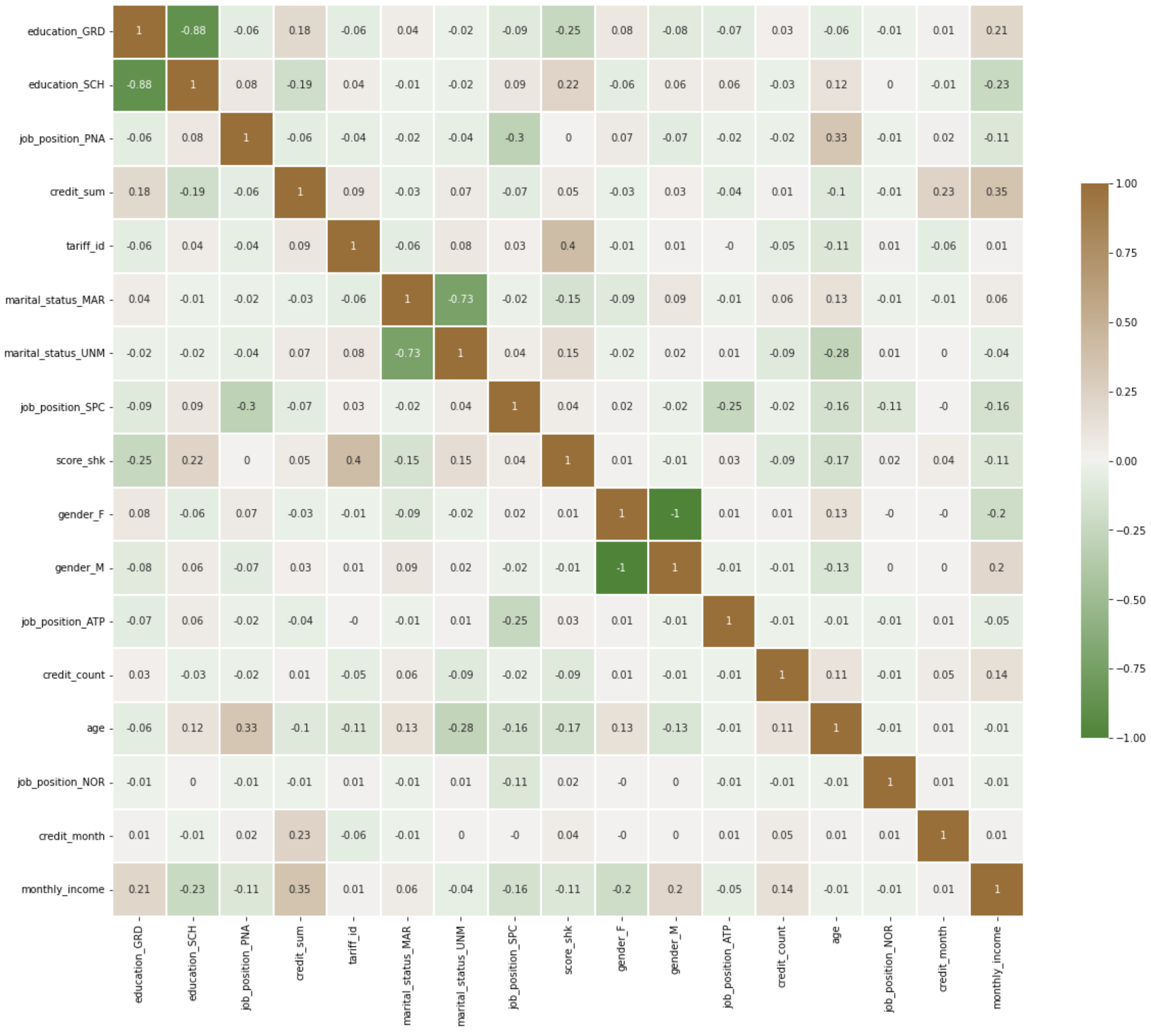
```
print('Признаки, имеющие максимальную по модулю корреляцию с целевым при
best_params  =  data.corr()['open_account_flg'].map(abs).sort_values(ascen
best_params = best_params[best_params.values > 0.02]
best_params
```

Out[8]:

```
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
```

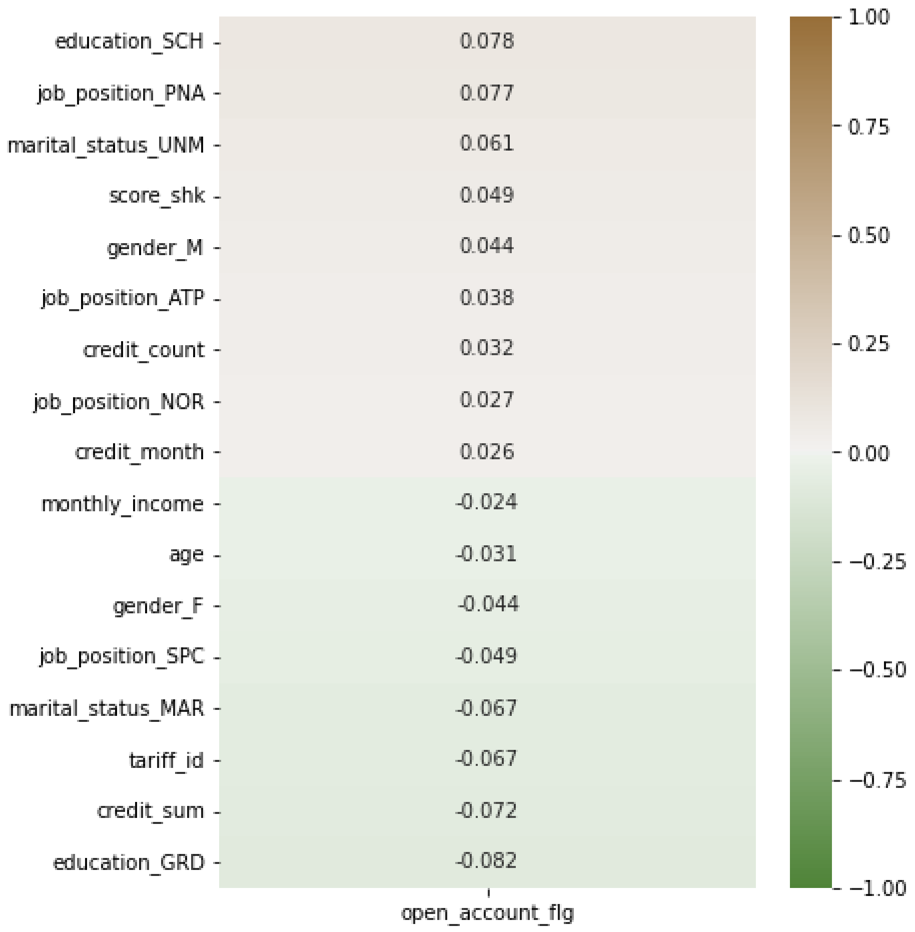
In [9]:

```
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()
```



2) Разделение выборки на обучающую и тестовую

In [11]:

```
data_best  =  data[best_params.index]
data_best.head()
```

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
```

3) 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	

4) Модель №1: Случайный лес

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)))
```

In [15]:

```
print_metrics(y_test, RandomForestClassifier(random_state=17).fit(x_trai
```

```
Precision: 0.4737991266375546
F1-measure: 0.15617128463476068
```

Подбор гиперпараметров

In [90]:

```
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_)
```

```
{'max_features': 'auto', 'min_samples_leaf': 5, 'n_estimators': 1000}
```

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)
```

5) Модель №2: Градиентный бустинг

In [16]:

```
print_metrics(y_test,  GradientBoostingClassifier(random_state=17).fit(x_
```

```
Accuracy:  0.8218803685772295
Precision:  0.5819672131147541
Recall:  0.06118052563550194
F1-measure:  0.11072124756335285
```

Подбор гиперпараметров

In [76]:

```
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_)
```

```
{'min_samples_leaf': 5, 'n_estimators': 200}
```

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)
```

```
Precision:  0.5709876543209876
F1-measure:  0.13988657844990549
```

6) Модель №3: Стекинг

In [24]:

```
dataset = Dataset(x_train, y_train, x_test)
```

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)),
                               ('rf_3',
                                GradientBoostingClassifier(n_estimators=2
00))],
                   final_estimator=StackingClassifier(estimators=[('dt_
2',
                                                                  Decisi
onTreeClassifier()),
                                                                 ('rf_
2',
                                                                  Random
ForestClassifier(n_estimators=10,

random_state=42)),
                                                                 ('rf_
4',
                                                                  Gradie
ntBoostingClassifier(n_estimators=20))],
                                                      final_estimator=MLP
Classifier(random_state=1488)))
```

In [98]:

```
print_metrics(y_test, clf.predict(x_test))
```

```
Precision:  0.6148148148148148
F1-measure:  0.12813585488228482
mean_absolute_error:  0.18
median_absolute_error:  0.0
r2_score: -0.19
```

Сравнение моделей

In [6]:

```
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)
```

```
Случайный лес
```

```
-------------------------------------------------------------------------
--
NameError                                Traceback (most recent call las
t)
~\AppData\Local\Temp/ipykernel_12728/287930607.py in <module>
      1 print("Случайный лес")
----> 2 print_metrics(y_test, y_pred_rf)
      3 
      4 print("\nГрадиентный бустинг")
      5 print_metrics(y_test, y_pred_gb)

NameError: name 'print_metrics' is not defined
```

In [ ]: