

МОДЕЛИРОВАНИЕ СКОРИНГОВОЙ СИСТЕМЫ БАНКА

Выполнил: Самсонов Марсель Васильевич

Руководитель: к.э.н. доцент Заграновская Анна Васильевна

Определение скоринга

Система оценки заемщика, с помощью которой банки и крупные микрофинансовые организации (МФО) могут предсказать, насколько аккуратно человек будет выплачивать кредит.

Актуальность

Скоринговые системы позволяют существенно сократить время на принятие решения о выдаче кредита или другой услуги



Цель работы

Смоделировать скоринговую систему банка

Задачи

1. Использовать традиционные методы в кредитном скоринге
2. Проанализировать влияние кластеризации на качество скоринговой системы
3. Применить нейронную сеть в кредитном скоринге
4. Сравнить построенные модели по расширенным критериям качества

Данные для анализа

статус-счета	срок-кредита	кред-история	цель	сумма-кредита	сбер-ния	прод-работы-у-работ-ля	платежи-в-проц	сем-й-статус	другие-обяз-ва	...	собст-ть	возраст	другие-рассрочки	тип-вла-я-жильем	число-кред	статус-работ-ка	кол-во-поруч-й	телефон	иностр-работник	кред-риск	
0	1	18	4	2	1049	1	2	4	2	1	...	2	21	3	1	1	3	2	1	2	1
1	1	9	4	0	2799	1	3	2	3	1	...	1	36	3	1	2	3	1	1	2	1
2	2	12	2	9	841	2	4	2	2	1	...	1	23	3	1	1	2	2	1	2	1
3	1	12	4	0	2122	1	3	3	3	1	...	1	39	3	1	2	2	1	1	1	1
4	1	12	4	0	2171	1	3	4	3	1	...	2	38	1	2	2	2	1	1	1	1
5	1	10	4	0	2241	1	2	1	3	1	...	1	48	3	1	2	2	1	1	1	1

1. статус счета
2. срок кредита
3. кредитная история
4. цель
5. сумма кредита
6. сбережения
7. продолжительность работы у работодателя
8. платежи в процентах
9. семейный статус
10. другие обязательства
11. постоянное проживание
12. собственность
13. возраст
14. другие рассрочки
15. тип владения жильем
16. число кредитов
17. статус работника
18. количество поручителей
19. телефон
20. иностранный работник
21. кредитный риск

Статистический анализ данных

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
#   Column                               Non-Null Count  Dtype
---  ---                               ---
0   статус-счета                         1000 non-null   int64
1   срок-кредита                         1000 non-null   int64
2   кред-история                         1000 non-null   int64
3   цель                                 1000 non-null   int64
4   сумма-кредита                       1000 non-null   int64
5   сбер-ния                             1000 non-null   int64
6   прод-работы-у-работ-ля             1000 non-null   int64
7   платежи-в-проц                     1000 non-null   int64
8   сем-й-статус                       1000 non-null   int64
9   другие-обяз-ва                     1000 non-null   int64
10  постоянное-проживание              1000 non-null   int64
11  собст-ть                            1000 non-null   int64
12  возраст                             1000 non-null   int64
13  другие-рассрочки                   1000 non-null   int64
14  тип-вла-я-жильем                   1000 non-null   int64
15  число-кред                          1000 non-null   int64
16  статус-работ-ка                     1000 non-null   int64
17  кол-во-поруч-й                     1000 non-null   int64
18  телефон                             1000 non-null   int64
19  иност-работник                      1000 non-null   int64
20  кред-риск                           1000 non-null   int64
dtypes: int64(21)
memory usage: 164.2 KB
```

```
df.groupby(['кред-риск']).size()
```

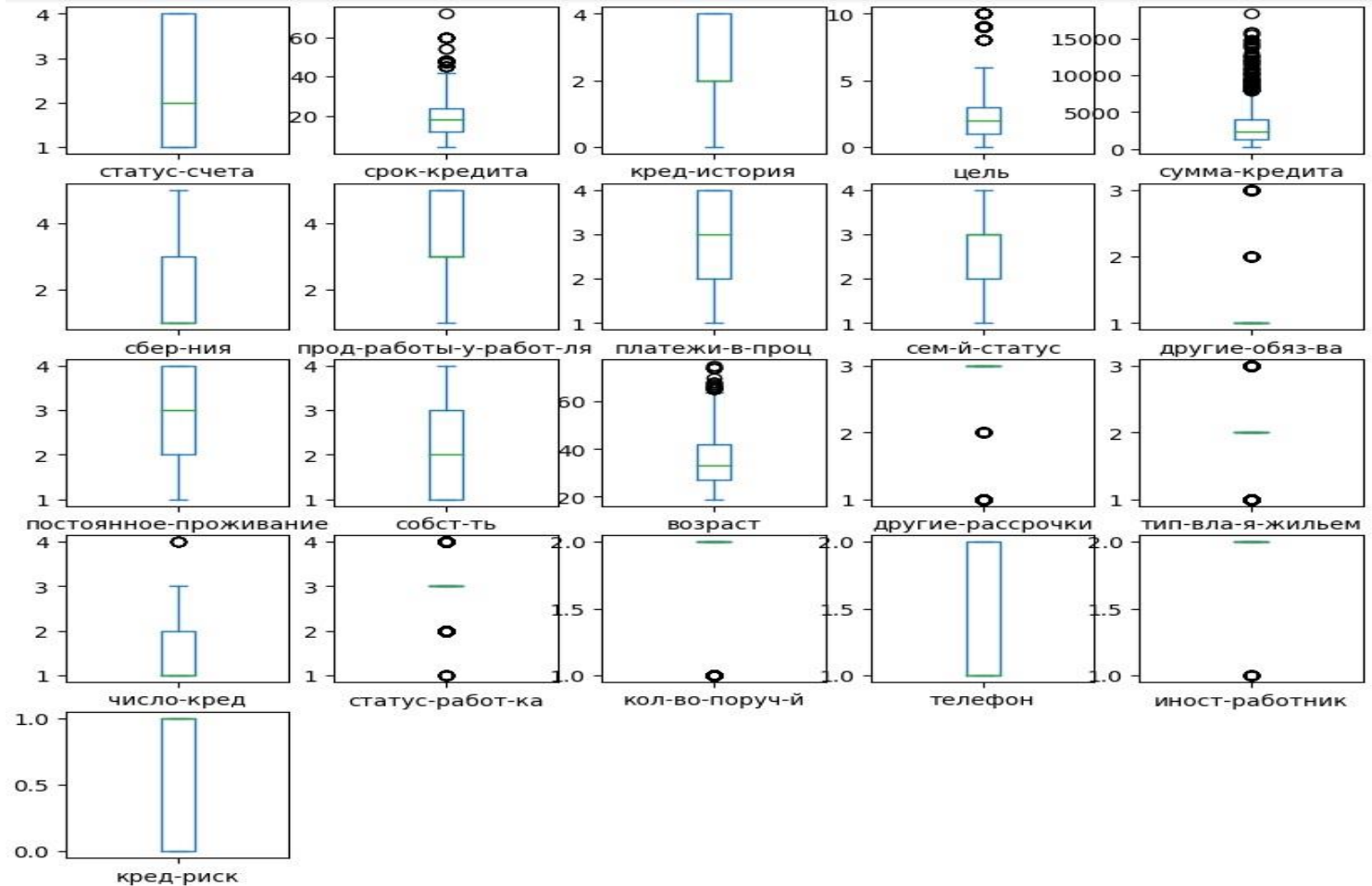
```
кред-риск
0      300
1      700
dtype: int64
```

```
df.describe()
```

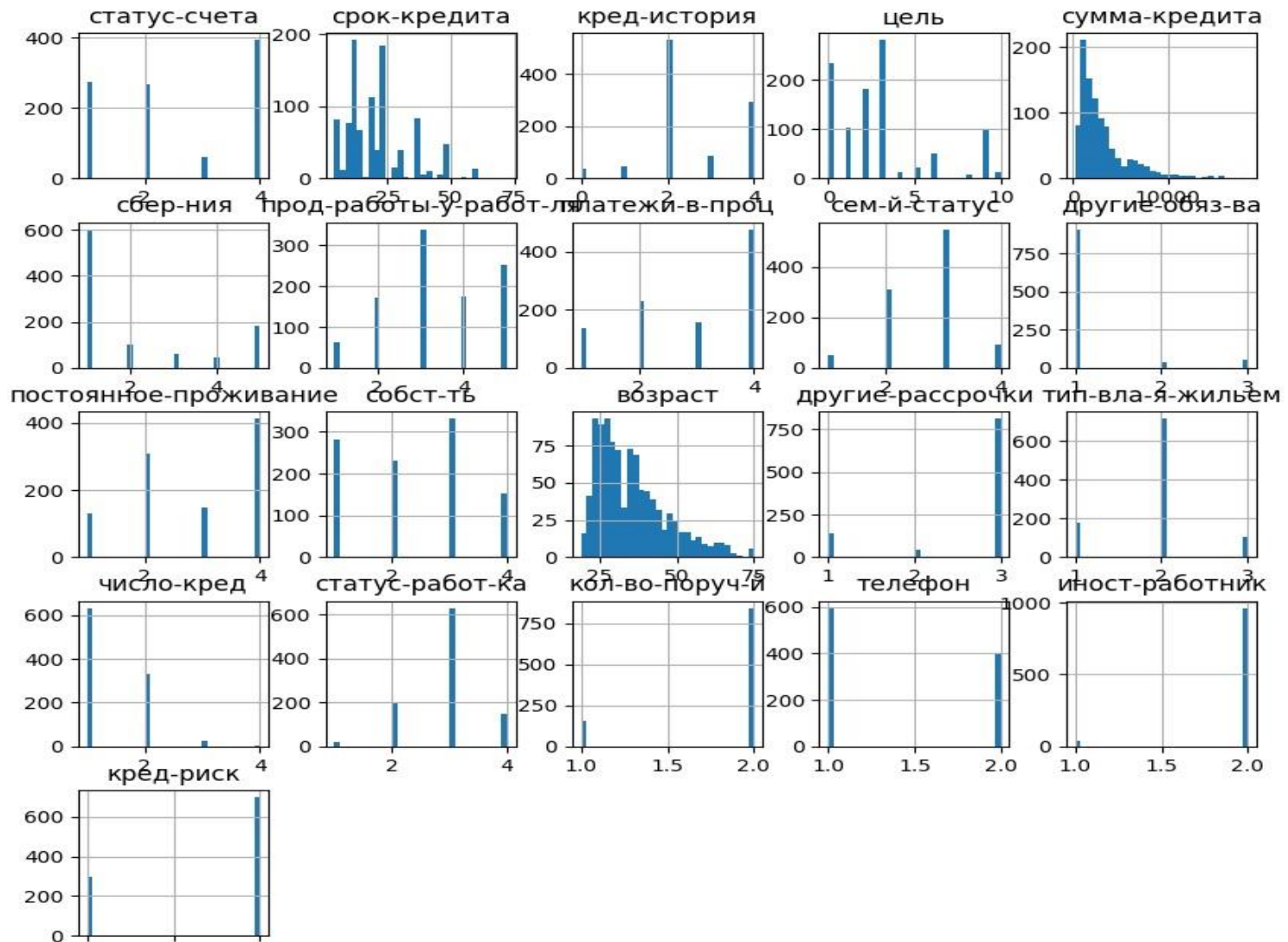
	статус-счета	срок-кредита	кред-история	цель	сумма-кредита	сбер-ния	прод-работы-у-работ-ля	платежи-в-проц	сем-й-статус	другие-обяз-ва	...	собст-ть	возраст	другие-рассрочки	тип-вла-я-жильем	число-кред	статус-работ-ка	кол-во-поруч-й
count	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	...	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000	1000.000
mean	2.577	20.903	2.545	2.828	3271.248	2.105	3.384	2.973	2.682	1.145	...	2.358	35.542	2.675	1.928	1.407	2.904	1.845
std	1.258	12.059	1.083	2.744	2822.752	1.580	1.208	1.119	0.708	0.478	...	1.050	11.353	0.706	0.530	0.578	0.654	0.362
min	1.000	4.000	0.000	0.000	250.000	1.000	1.000	1.000	1.000	1.000	...	1.000	19.000	1.000	1.000	1.000	1.000	1.000
25%	1.000	12.000	2.000	1.000	1365.500	1.000	3.000	2.000	2.000	1.000	...	1.000	27.000	3.000	2.000	1.000	3.000	2.000
50%	2.000	18.000	2.000	2.000	2319.500	1.000	3.000	3.000	3.000	1.000	...	2.000	33.000	3.000	2.000	1.000	3.000	2.000
75%	4.000	24.000	4.000	3.000	3972.250	3.000	5.000	4.000	3.000	1.000	...	3.000	42.000	3.000	2.000	2.000	3.000	2.000
max	4.000	72.000	4.000	10.000	18424.000	5.000	5.000	4.000	4.000	3.000	...	4.000	75.000	3.000	3.000	4.000	4.000	2.000

Визуальный анализ данных

Box-plot



Гистограммы



Корректировка несбалансированной выборки результативного признака

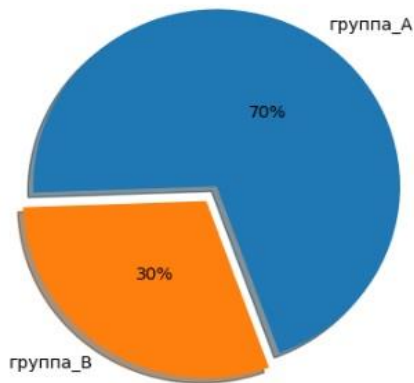


Рис 1. Распределение данных до корректировки

```
oversample = ADASYN()  
X,Y = oversample.fit_resample(X,Y)
```

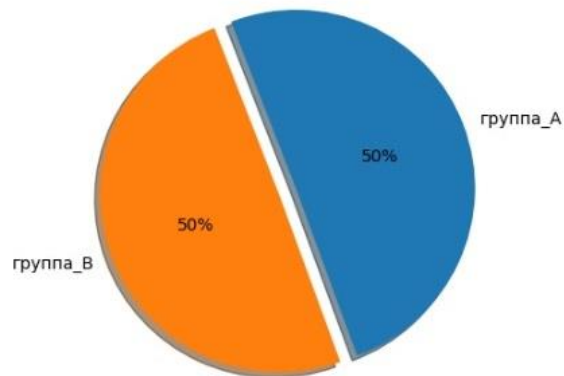


Рис 2. Распределение данных после корректировки

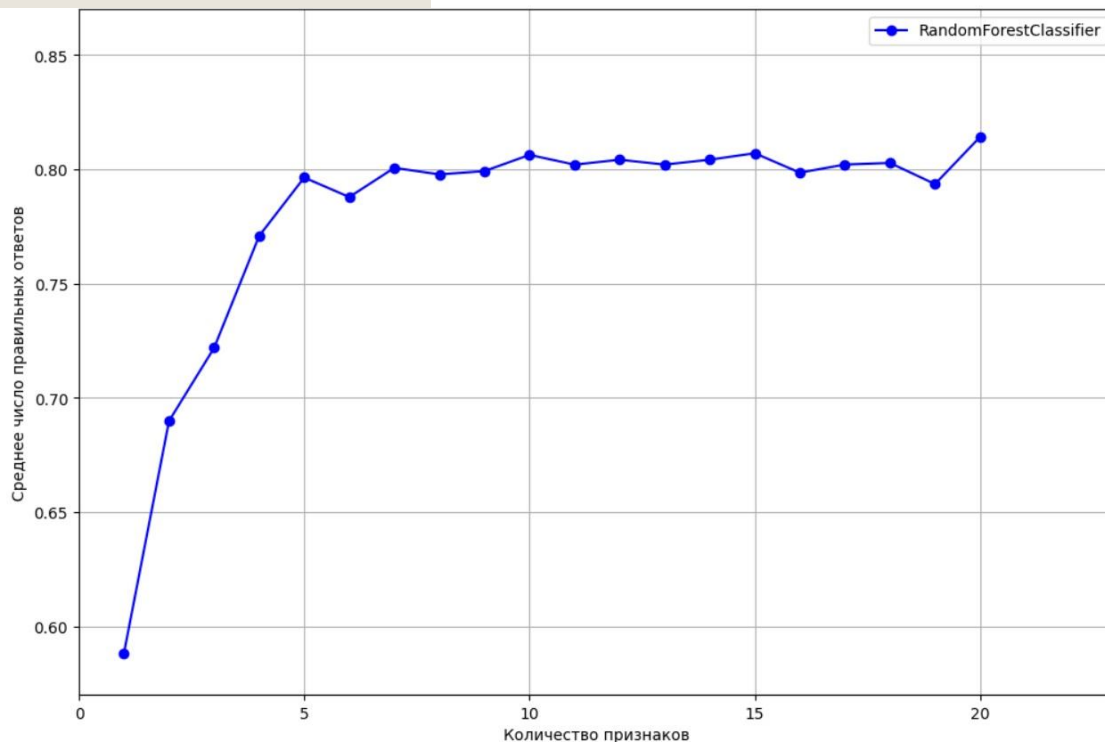
Масштабирование

```
scaler = MinMaxScaler(feature_range=(0,10))  
scaler.fit_transform(X)
```

	статус-счета	срок-кредита	кред-история	цель	сумма-кредита	сбер-ния	прод-работы-у-работ-ля	платежи-в-проц	сем-й-статус	другие-обяз-ва	постоянное-проживание	собст-ть	возраст	другие-рассрочки	тип-вла-я-жильем	число-кред	статус-работ-ка	кол-во-поруч-й
0	0.000000	2.058824	10.0	2.0	0.439639	0.0	2.5	10.000000	3.333333	0.0	10.000000	3.333333	0.357143	10.0	0.0	0.000000	6.666667	10.0
1	0.000000	0.735294	10.0	0.0	1.402553	0.0	5.0	3.333333	6.666667	0.0	3.333333	0.000000	3.035714	10.0	0.0	3.333333	6.666667	0.0
2	3.333333	1.176471	5.0	9.0	0.325190	2.5	7.5	3.333333	3.333333	0.0	10.000000	0.000000	0.714286	10.0	0.0	0.000000	3.333333	10.0
3	0.000000	1.176471	10.0	0.0	1.030043	0.0	5.0	6.666667	6.666667	0.0	3.333333	0.000000	3.571429	10.0	0.0	3.333333	3.333333	0.0
4	0.000000	1.176471	10.0	0.0	1.057005	0.0	5.0	10.000000	6.666667	0.0	10.000000	3.333333	3.392857	0.0	5.0	3.333333	3.333333	10.0
...
1404	0.000000	2.941176	5.0	3.0	0.978321	0.0	2.5	0.000000	6.666667	0.0	0.000000	0.000000	0.357143	10.0	0.0	0.000000	3.333333	0.0

Отбор факторных признаков

```
model=RandomForestClassifier()
for i in lst_f:
    rfe=RFE(model,n_features_to_select=i)
    fit=rfe.fit(X,Y)
    features=fit.transform(X)
    results=cross_val_score(model, features,Y, cv=7)
    res = results.mean()
    k.append(res)
```



ИСПОЛЬЗОВАНИЕ ТРАДИЦИОННЫХ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ В КРЕДИТНОМ СКОРИНГЕ

Линейные модели:

LR – логистическая регрессия

SVM – линейный метод опорных векторов

LDA – линейный дискриминантный анализ

Нелинейные модели:

KNN – модель ближайшего соседа

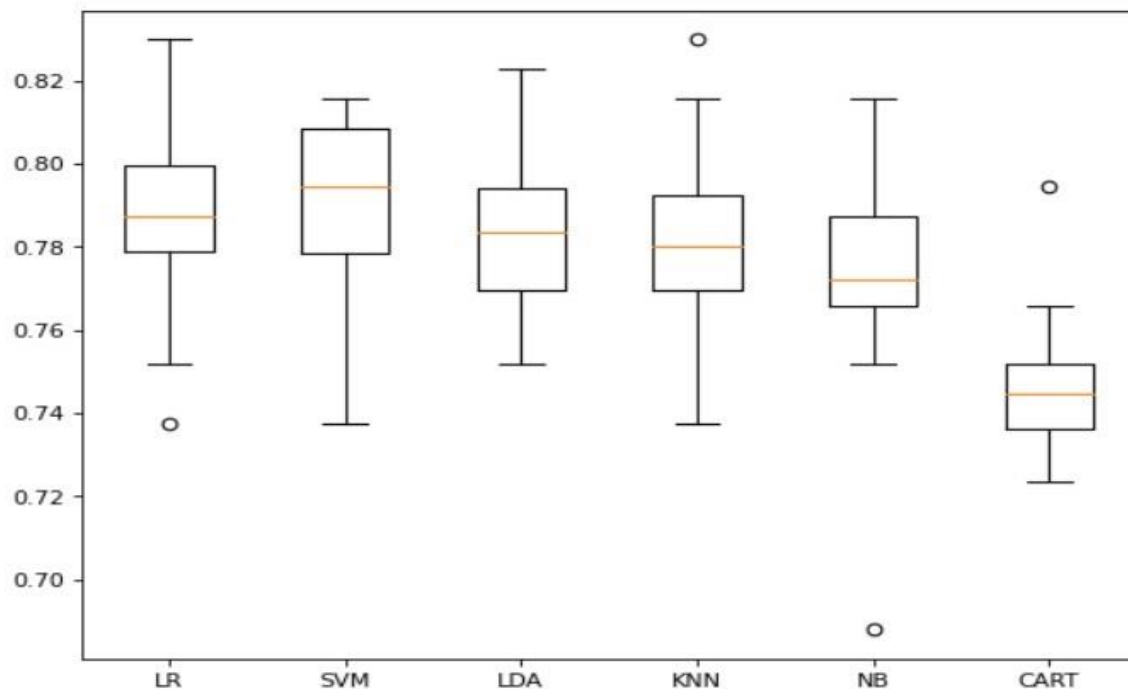
NB – наивный байесовский классификатор

CART – модель решающего дерева

Выбор лучшей модели

```
scoring = 'accuracy'  
for name, model in models:  
    kfold = KFold(n_splits= 10, random_state = 7, shuffle=True)  
    cv_results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring)  
    results.append(cv_results)  
    names.append(name)  
    print(name, cv_results.mean(), cv_results.std())
```

ACCURACY : SVM 0.79



Прогнозирование на основе лучшей модели по всем данным

```
name, model_svc = models[1]
model_svc = SVC(kernel='linear')
model_svc.fit(X_train, Y_train)
predictions = model_svc.predict(X_test)
accuracy_score(Y_test, predictions)
```

Матрица ошибок

```
confusion_matrix(Y_test, predictions)
array([[106, 25],
       [ 38, 113]])
```

Доля правильных ответов : 77%

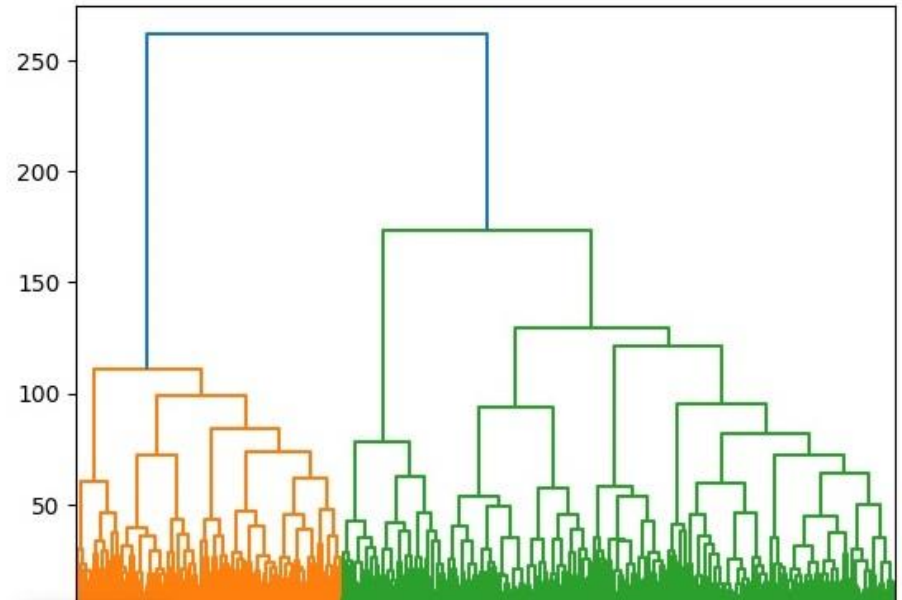
Оценка прогнозирования выбранной модели по расширенным метрикам качества

```
report = classification_report(Y_test, predictions)
print(report)
```

	precision	recall	f1-score	support
0	0.74	0.81	0.77	131
1	0.82	0.75	0.78	151
accuracy			0.78	282
macro avg	0.78	0.78	0.78	282
weighted avg	0.78	0.78	0.78	282

Кластеризация клиентов для последующей классификации по каждому кластеру

```
link = linkage(df_xy, 'ward', 'euclidean')
dn = dendrogram(link)
df_xy['cluster'] = fcluster(link, 2, criterion = 'maxclust')
mask1 = df_xy['cluster'] == 1
mask2 = df_xy['cluster'] == 2
df_xy1 = df_xy[mask1]
df_xy2 = df_xy[mask2]
df_xy.groupby('cluster').mean()
```



кластер	статус-счета	срок-кредита	кред-история	цель	сумма-кредита	сбер-ния	прод-работы-у-работ-ля	платежи-в-проц	сем-й-статус	другие-обяз-ва	...	собст-ть	возраст	другие-рассрочки	тип-вла-я-жильем
1.0	3.974085	2.954008	5.650709	3.455137	2.518813	2.678305	5.914438	5.538133	5.377467	0.867157	...	4.665848	3.204163	6.570305	4.348156
2.0	3.848065	2.445812	5.654952	2.571885	1.358311	2.012780	5.362087	6.517572	5.069223	0.623003	...	3.851615	2.646242	7.896699	3.956337

Коррекция дисбаланса кластеров

```
df_xy.groupby(['cluster']).size()
```

До коррекции:

Cluster

1 470

2 939

После коррекции:

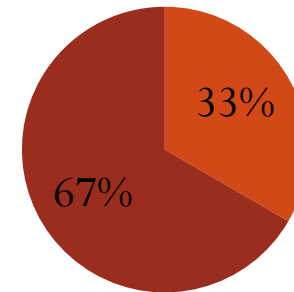
Cluster

1 946

2 939

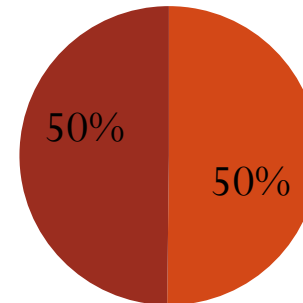
До

■ Кластер 1 ■ Кластер 2



После

■ Кластер 1 ■ Кластер 2



Классификация по каждому кластеру линейным методом опорных векторов

```
array_0 = df_xy1.values
X0 = array_0[:, :20]
Y0 = array_0[:, 20]
array_1 = df_xy2.values
X1 = array_1[:, :20]
Y1 = array_1[:, 20]
X0_train, X0_test, Y0_train, Y0_test = train_test_split(X0, Y0, test_size=0.2, random_state=7)
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y1, test_size=0.2, random_state=7)
model_0 = SVC(kernel='linear')
model_1 = SVC(kernel='linear')
model_0.fit(X0_train, Y0_train)
model_1.fit(X1_train, Y1_train)
predictions_0 = model_0.predict(X0_test)
predictions_1 = model_1.predict(X1_test)
accuracy_score(Y0_test, predictions_0)
accuracy_score(Y1_test, predictions_1)
```

Доля правильных
ответов (SVM):

Кластер 1 - 83%

Кластер 2 - 79%

Оценка прогнозирования по кластерам расширенными метриками качества

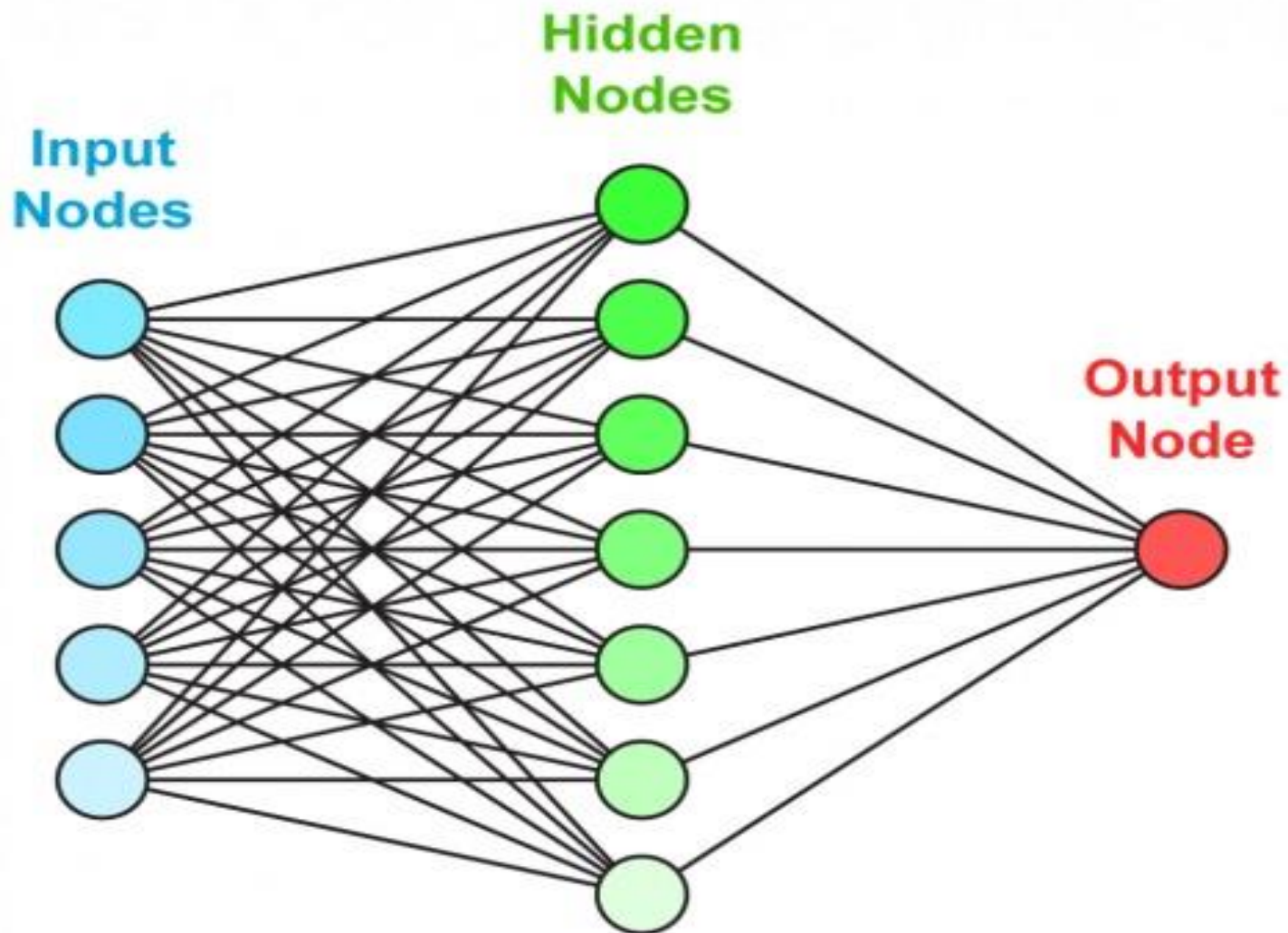
Кластер 1

	precision	recall	f1-score	support
0	0.85	0.86	0.85	108
1	0.81	0.79	0.80	82
accuracy			0.83	190
macro avg	0.83	0.83	0.83	190
weighted avg	0.83	0.83	0.83	190

Кластер 2

	precision	recall	f1-score	support
0	0.85	0.81	0.83	114
1	0.72	0.78	0.75	74
accuracy			0.80	188
macro avg	0.79	0.80	0.79	188
weighted avg	0.80	0.80	0.80	188

НЕЙРОННАЯ СЕТЬ В КРЕДИТНОМ СКОРИНГЕ



Архитектура нейронной сети

```
model = Sequential()  
model.add(Dense(9, input_dim=20, activation='relu'))  
model.add(Dense(10, activation='relu', ))  
model.add(Dense(10, activation='relu', ))  
model.add(Dense(10, activation='relu', ))  
model.add(Dense(10, activation='relu', ))  
model.add(Dense(2, activation='softmax'))
```

Оценка модели

```
scores = model.evaluate(X_test, y_test_bin)
```

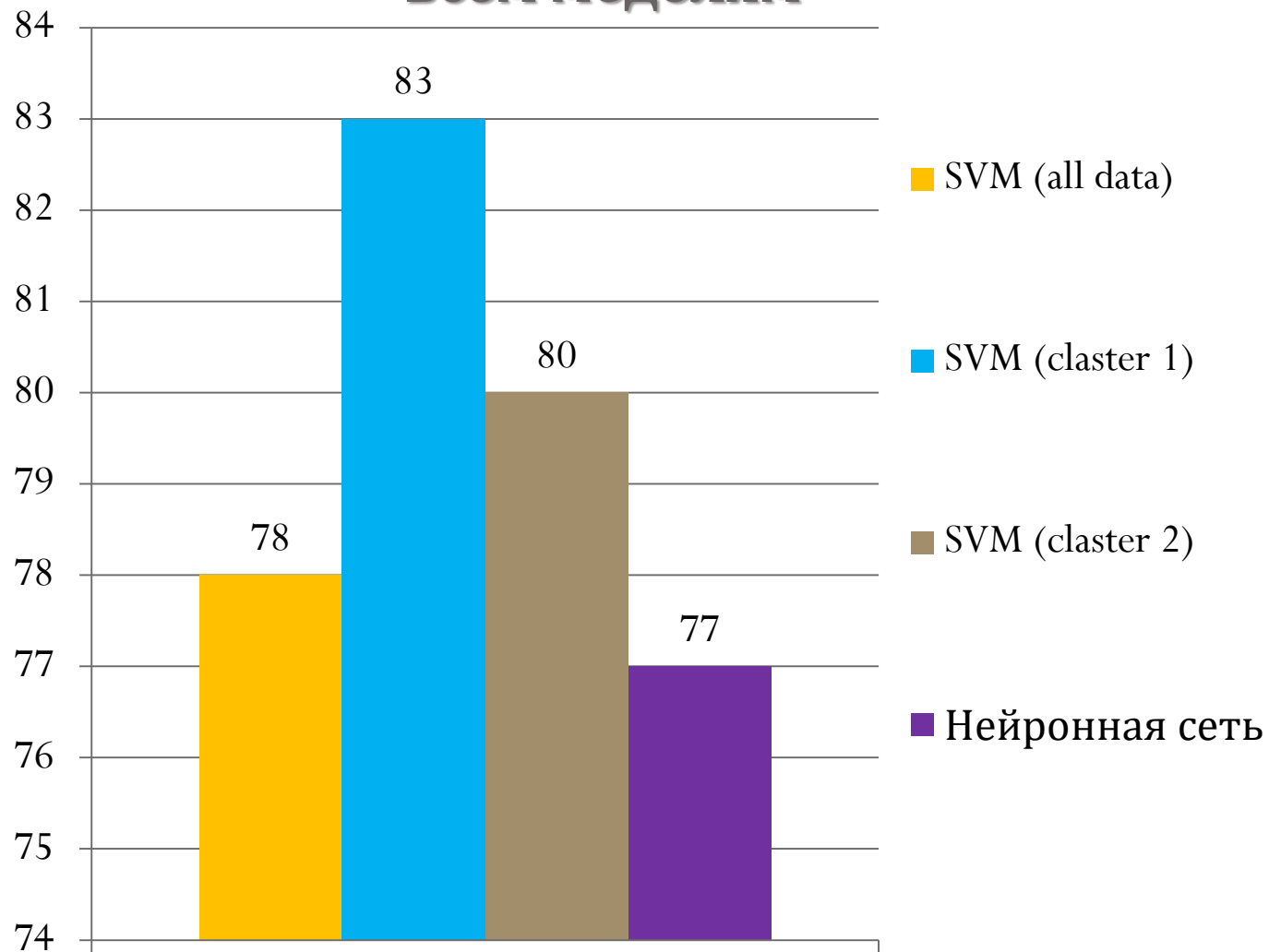
Доля правильных ответов: 78%

Оценка прогнозирования нейронной сети по расширенным метрикам качества

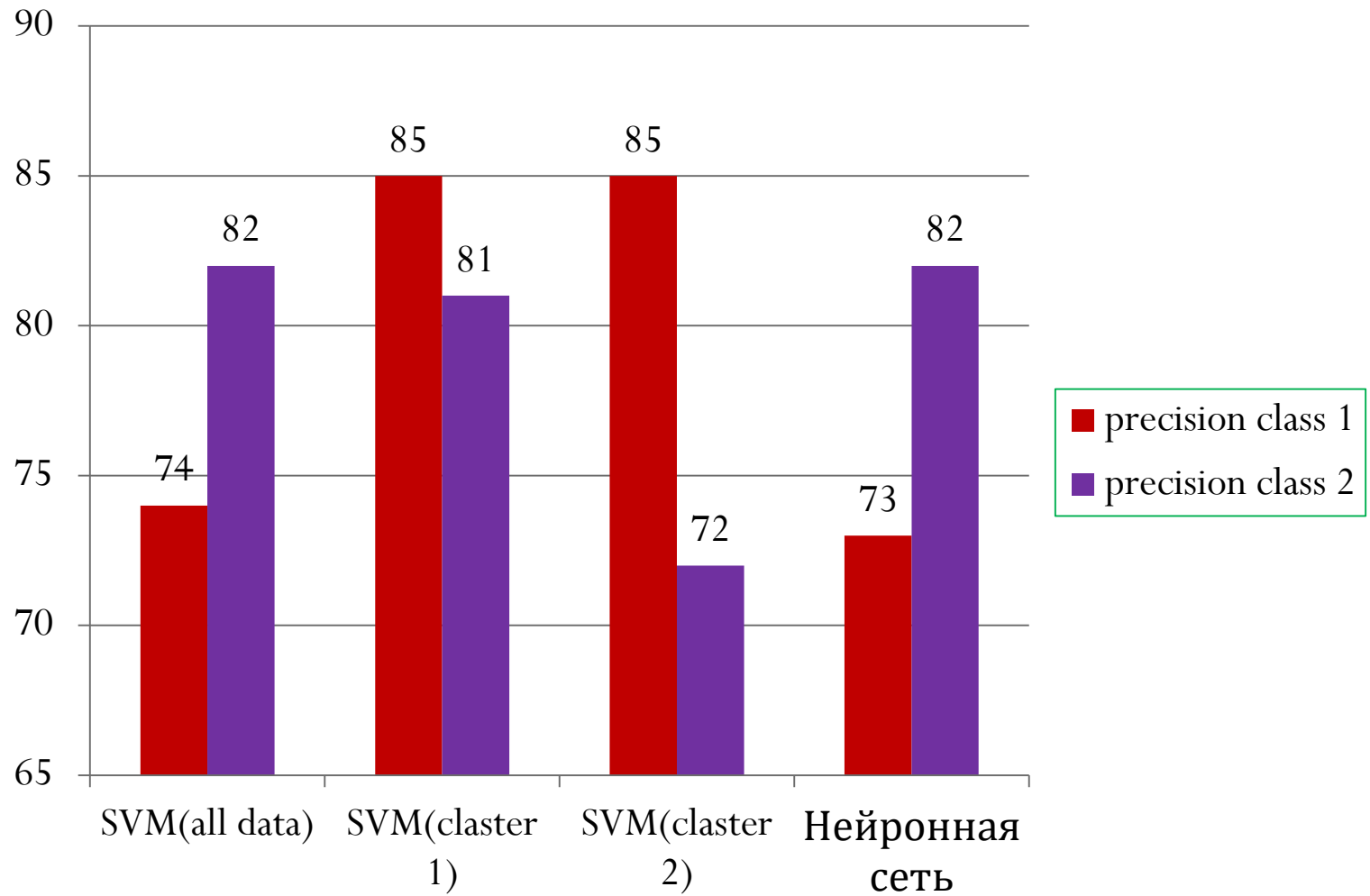
```
report = classification_report(Y_test, y_arr)
print(report)
```

	precision	recall	f1-score	support
0	0.73	0.81	0.77	131
1	0.82	0.74	0.77	151
accuracy			0.77	282
macro avg	0.77	0.77	0.77	282
weighted avg	0.77	0.77	0.77	282

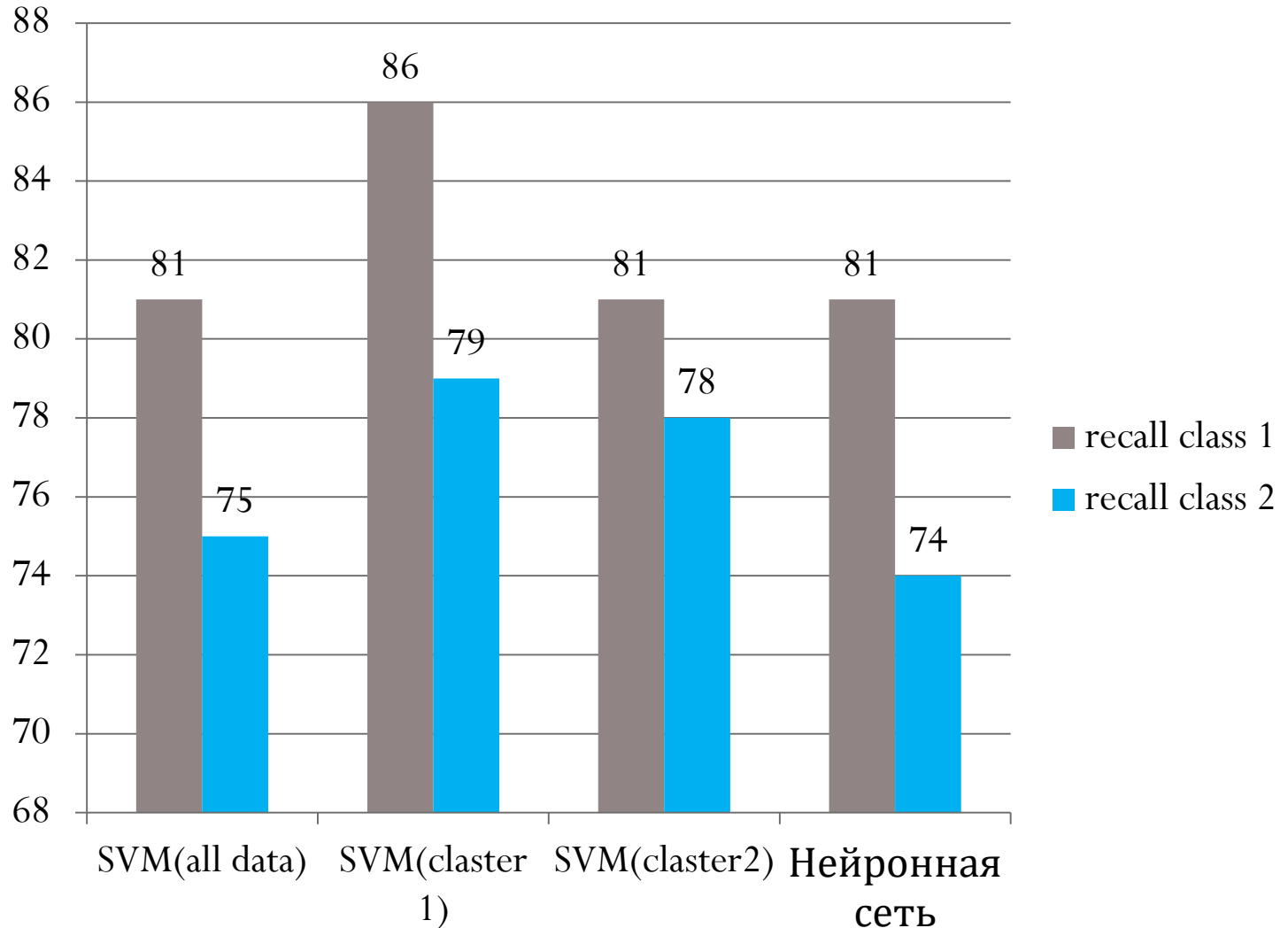
Доля правильных ответов(ассигасу) по всем моделям



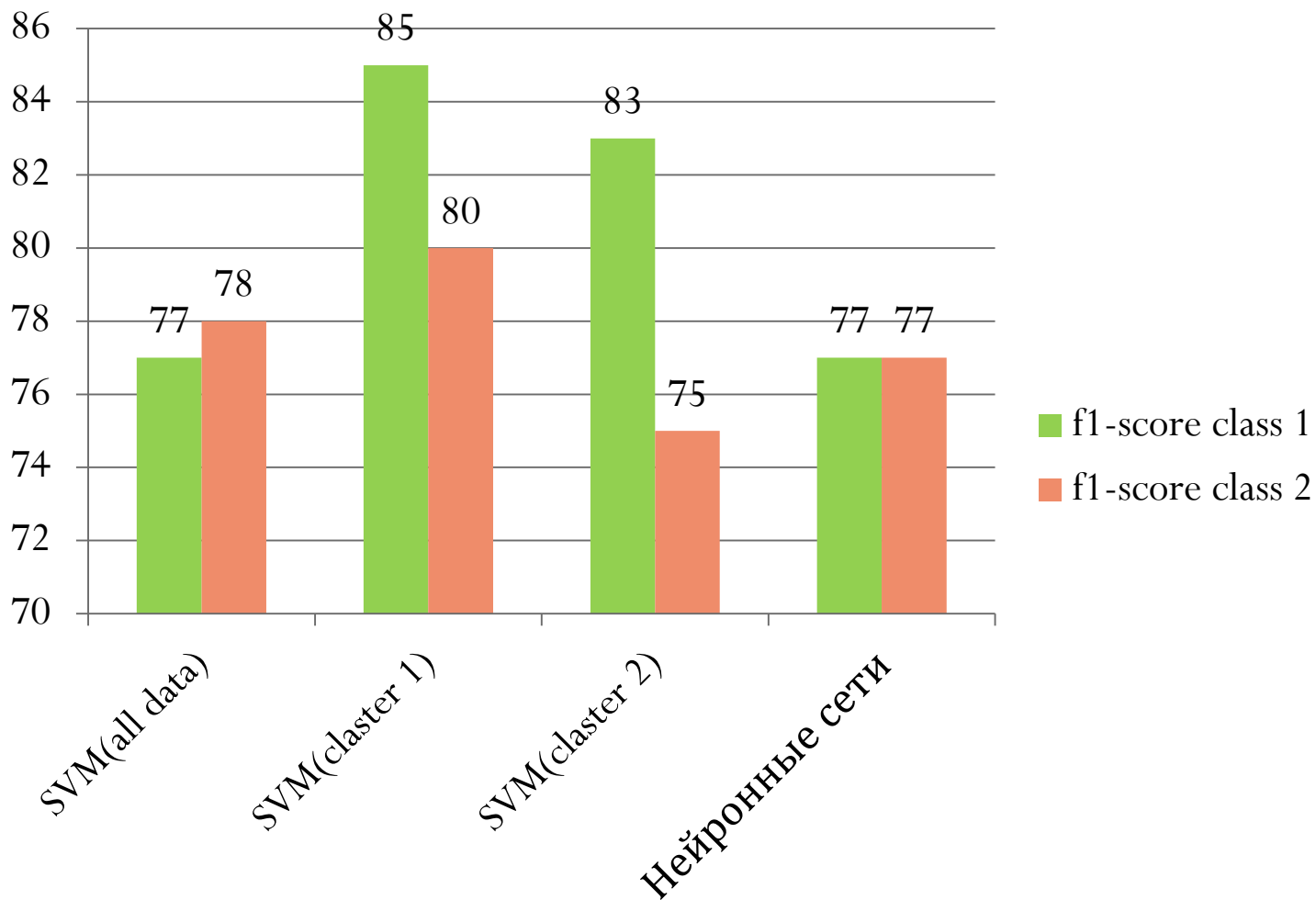
Точность (precision) по всем моделям



Полнота(recall) по всем моделям



Ф-мера(f1-score) по всем моделям



Сводная таблица расширенных критериев качества (macro avg) всех моделей

	SVM(all data)	SVM(cluster1)	SVM(cluster2)	NNET
Accuracy	0.78	0.83	0.80	0.77
Precision	0.78	0.83	0.79	0.77
Recall	0.78	0.83	0.80	0.77
F-score	0.78	0.83	0.79	0.77

ВЫВОД

- Построены модели скоринговых систем с применением трех подходов:
 1. Классическая модель на основе всех данных
 2. Классическая модель с группировкой данных по кластерам
 3. Нейронная сеть классификации
- Критерии качества созданных моделей приемлемы для прогнозирования в банковском скоринге.
- Сравнивая полученные результаты можно сказать, что подход с разделением данных на кластеры улучшает качество прогноза.
- Классические модели кредитного скоринга по качеству прогнозирования сопоставимы с системами скоринга в основе которых лежит нейронная сеть последовательного типа.