ПРИЛОЖЕНИЕ И
ROC анализ данных для модели

Таблица И.1 – Оценки классифицирующей способности отдельных предикторов

Показатель	AUC	KS	p-значение	Пропуски	Пропуски Нет дефолта%	Пропуски Дефолт%
1	2	3	4	5	6	7
Автомобиль год выпуска1	0,572	0,121	0,000	195030	96,972	3,028
Автомобиль год выпуска2	0,613	0,239	0,000	234900	96,935	3,065
Автомобиль год выпуска3	0,768	0,530	0,013	236396	96,924	3,076
Воинская служба	0,669	0,301	0,000	108124	98,884	1,116
Количество детей	0,511	0,023	0,092	155538	97,262	2,738
Количество иждивенцев	0,552	0,106	0,003	227405	96,927	3,073
Недвижимость	0,526	0,052	0,000	0	0,000	0,000
Образование	0,666	0,254	0,000	70250	99,162	0,838
Отношение к банку	0,505	0,010	0,480	0	0,000	0,000
Работа занимаемая должность	0,649	0,284	0,000	0	0,000	0,000
Работа последнее место стаж лет	0,500	0,001	1,000	217748	97,149	2,851
Работа уровень дохода BYR	0,507	0,018	1,000	221099	96,816	3,184
Семейное положение	0,653	0,216	0,000	63143	99,012	0,988
Собственная квартира	0,513	0,025	0,000	0	0,000	0,000
Собственный дом	0,511	0,021	0,003	0	0,000	0,000
Уголовная ответственность	0,516	0,031	0,000	0	0,000	0,000
Адрес проживания - Населенный пункт	0,538	0,072	0,000	0	0,000	0,000
Адрес проживания - Тип населенного пункта	0,621	0,222	0,000	80329	98,873	1,127
Гражданин РБ	0,593	0,184	0,000	68942	98,808	1,192
Дата рождения	0,560	0,095	0,000	0	0,000	0,000
Был ли хоть один договор прекращен досрочно	0,606	0,187	0,000	134681	97,903	2,097

Окончание таблицы И.1

1	2	3	4	5	6	7
Количество действующих договоров обеспечения	0,512	0,028	0,073	173674	97,070	2,930
Количество действующих кредитных договоров	0,559	0,072	0,000	145639	97,743	2,257
Количество запросов в КБ за последние 30 дней	0,570	0,109	0,000	132363	97,989	2,011
Количество фактов просрочки по основному долгу	0,650	0,242	0,000	168714	97,274	2,726
Максимальное количество дней просрочки	0,652	0,257	0,000	173510	97,238	2,762
Максимальный срок, на который заключался договор, в годах	0,539	0,083	0,000	135529	97,886	2,114
Наличие кредитной истории	0,600	0,201	0,000	131648	97,998	2,002
Общее количество запросов в КБ	0,537	0,074	0,000	136244	97,769	2,231
Сумма кредитных лимитов	0,544	0,078	0,000	153215	97,637	2,363
Сумма договора	0,651	0,275	0,000	20133	98,425	1,575
Количество потребляемых банковских продуктов	0,590	0,168	0,000	117790	97,925	2,075
Пол	0,542	0,084	0,000	0	0,000	0,000
Социальная группа	0,615	0,210	0,000	89220	98,525	1,475
Код подразделения	0,689	0,304	0,000	0	0,000	0,000
Автомобиль год выпуска	0,571	0,119	0,000	194937	96,974	3,026
Число авто	0,508	0,015	0,100	0	0,000	0,000
Есть авто	0,507	0,015	0,100	0	0,000	0,000
Срок кредита в днях	0,635	0,283	0,000	1	100,000	0,000
Ежедневный платеж	0,538	0,083	0,000	12330	97,405	2,595
Столица	0,522	0,044	0,000	0	0,000	0,000
Областной центр	0,529	0,059	0,000	0	0,000	0,000

Примечание – Источник: собственная разработка.

Таблица И.2 – Оценки классифицирующей численных предикторов и их производных

Показатель	AUC	KS	p-значение	Пропуски	Пропуски Нет дефолта%	Пропуски Дефолт%
1	2	3	4	5	6	7
Автомобиль год выпуска1	0,572	0,121	0,000	195030	96,972	3,028
Автомобиль год выпуска1 (бинарный)	0,507	0,014	0,134	0	0,000	0,000
Автомобиль год выпуска1 (пропуск => 0)	0,501	0,015	0,084	0	0,000	0,000
Автомобиль год выпуска1 (разбивка по KS)	0,520	0,029	0,000	195030	96,972	3,028
Автомобиль год выпуска2	0,613	0,239	0,000	234900	96,935	3,065
Автомобиль год выпуска2 (бинарный)	0,502	0,005	0,999	0	0,000	0,000
Автомобиль год выпуска2 (пропуск => 0)	0,502	0,004	1,000	0	0,000	0,000
Автомобиль год выпуска2 (разбивка по KS)	0,502	0,005	0,999	234900	96,935	3,065
Автомобиль год выпуска3	0,768	0,530	0,013	236396	96,924	3,076
Автомобиль год выпуска3 (бинарный)	0,500	0,001	1,000	0	0,000	0,000
Автомобиль год выпуска3 (пропуск => 0)	0,500	0,000	1,000	0	0,000	0,000
Автомобиль год выпуска3 (разбивка по KS)	0,501	0,001	1,000	236396	96,924	3,076
Количество действующих кредитных договоров	0,559	0,072	0,000	145639	97,743	2,257

Продолжение таблицы И.2

1	2	3	4	5	6	7
Количество действующих кредитных договоров (бинарный)	0,585	0,170	0,000	0	0,000	0,000
Количество действующих кредитных договоров (пропуск => 0)	0,598	0,184	0,000	0	0,000	0,000
Количество действующих кредитных договоров (разбивка по KS)	0,592	0,170	0,000	145639	97,743	2,257
Количество запросов в КБ за последние 30 дней	0,570	0,109	0,000	132363	97,989	2,011
Количество запросов в КБ за последние 30 дней (бинарный)	0,600	0,200	0,000	0	0,000	0,000
Количество запросов в КБ за последние 30 дней (пропуск => 0)	0,580	0,150	0,000	0	0,000	0,000
Количество запросов в КБ за последние 30 дней (разбивка по KS)	0,612	0,200	0,000	132363	97,989	2,011
Количество иждивенцев	0,552	0,106	0,003	227405	96,927	3,073
Количество иждивенцев (бинарный)	0,501	0,002	1,000	0	0,000	0,000
Количество иждивенцев (пропуск => 0)	0,501	0,004	1,000	0	0,000	0,000
Количество иждивенцев (разбивка по KS)	0,502	0,003	1,000	227405	96,927	3,073

Продолжение таблицы И.2

1	2	3	4	5	6	7
Количество фактов просрочки по основному долгу	0,650	0,242	0,000	168714	97,274	2,726
Количество фактов просрочки по основному долгу(бинарный)	0,542	0,084	0,000	0	0,000	0,000
Количество фактов просрочки по основному долгу (пропуск => 0)	0,565	0,120	0,000	0	0,000	0,000
Количество фактов просрочки по основному долгу (разбивка по KS)	0,555	0,110	0,000	168714	97,274	2,726
Максимальное количество дней просрочки	0,652	0,257	0,000	173510	97,238	2,762
Максимальное количество дней просрочки (бинарный)	0,539	0,078	0,000	0	0,000	0,000
Максимальное количество дней просрочки (пропуск => 0)	0,566	0,126	0,000	0	0,000	0,000
Максимальное количество дней просрочки (разбивка по KS)	0,539	0,078	0,000	173510	97,238	2,762
Максимальный срок, на который заключался договор, в годах	0,539	0,083	0,000	135529	97,886	2,114
Максимальный срок, на который заключался договор, в годах(бинарный)	0,593	0,185	0,000	0	0,000	0,000

Продолжение таблицы И.2

1	2	3	4	5	6	7
Максимальный срок, на который заключался договор, в годах (пропуск => 0)	0,580	0,182	0,000	0	0,000	0,000
Максимальный срок, на который заключался договор, в годах (разбивка по KS)	0,603	0,185	0,000	135529	97,886	2,114
Общее количество запросов в КБ	0,537	0,074	0,000	136244	97,769	2,231
Общее количество запросов в КБ(бинарный)	0,582	0,164	0,000	0	0,000	0,000
Общее количество запросов в КБ(пропуск => 0)	0,586	0,151	0,000	0	0,000	0,000
Общее количество запросов в КБ(разбивка по KS)	0,590	0,164	0,000	136244	97,769	2,231
Срок кредита в днях	0,635	0,283	0,000	1	100,000	0,000
Срок кредита в днях(бинарный)	0,500	0,000	1,000	0	0,000	0,000
Срок кредита в днях (пропуск => 0)	0,635	0,283	0,000	0	0,000	0,000
Срок кредита в днях (разбивка по KS)	0,641	0,283	0,000	1	100,000	0,000
Сумма договора	0,651	0,275	0,000	20133	98,425	1,575
Сумма договора (бинарный)	0,521	0,043	0,000	0	0,000	0,000
Сумма договора (пропуск => 0)	0,653	0,286	0,000	0	0,000	0,000
Сумма договора (разбивка по KS)	0,631	0,262	0,000	20133	98,425	1,575

Окончание таблицы И.2

1	2	3	4	5	6	7
Сумма кредитных лимитов	0,544	0,078	0,000	153215	97,637	2,363
Сумма кредитных лимитов (бинарный)	0,578	0,155	0,000	0	0,000	0,000
Сумма кредитных лимитов (пропуск => 0)	0,588	0,173	0,000	0	0,000	0,000
Сумма кредитных лимитов (разбивка по KS)	0,584	0,166	0,000	153215	97,637	2,363

Примечание – Источник: собственная разработка.

Таблица И.3 – Оценки классифицирующей способности показателей вошедших на этап моделирования

Показатель	AUC	KS	p-значение	Пропуски	Пропуски Нет дефолта%	Пропуски Дефолт%
1	2	3	4	5	6	7
Воинская служба	0,651	0,210	0,000	0	0	0
Недвижимость	0,530	0,059	0,000	0	0	0
Образование	0,671	0,259	0,000	0	0	0
Работа занимаемая должность	0,653	0,294	0,000	0	0	0
Семейное положение	0,661	0,224	0,000	0	0	0
Собственная квартира	0,514	0,029	0,000	0	0	0
Собственный дом	0,511	0,021	0,005	0	0	0
Уголовная ответственность	0,516	0,033	0,000	0	0	0
Адрес проживания - Тип населенного пункта	0,633	0,249	0,000	0	0	0
Гражданин РБ	0,604	0,207	0,000	0	0	0
Дата рождения	0,566	0,101	0,000	0	0	0

Продолжение таблицы И.3

1	2	3	4	5	6	7
Был ли хоть один договор прекращен досрочно	0,619	0,213	0,000	0	0	0
Наличие кредитной истории	0,614	0,228	0,000	0	0	0
Сумма договора	0,651	0,275	0,000	0	0	0
Количество потребляемых банковских продуктов	0,596	0,176	0,000	0	0	0
Пол	0,543	0,085	0,000	0	0	0
Социальная группа	0,628	0,237	0,000	0	0	0
Код подразделения	0,705	0,337	0,000	0	0	0
Количество действующих кредитных договоров	0,602	0,196	0,000	0	0	0
Количество фактов просрочки по основному долгу	0,568	0,126	0,000	0	0	0
Максимальное количество дней просрочки	0,568	0,130	0,000	0	0	0
Сумма кредитных лимитов	0,592	0,184	0,000	0	0	0
Автомобиль год выпуска1	0,516	0,032	0,000	0	0	0
Количество запросов в КБ за последние 30 дней	0,625	0,227	0,000	0	0	0
Максимальный срок, на который заключался договор, в годах	0,615	0,211	0,000	0	0	0

Окончание таблицы И.3

1	2	3	4	5	6	7
Общее количество запросов в КБ	0,602	0,192	0,000	0	0	0
Срок кредита в днях	0,661	0,322	0,000	0	0	0

Примечание – Источник: собственная разработка.

ПРИЛОЖЕНИЕ К
Алгоритм оценки параметров

```
import torch.nn as nn
class ResultNet(nn.Module):
   '''Нейронная сеть в общем виде'''
   def __init__(self, neur_counts):
      super(ResultNet, self).__init__()
      # предполагаяется что выходной нейрон
      # всего один и пользователю не надо
      # его объявлять
      neur_counts = neur_counts.copy() + [1]
      # динамическое
      # добавление слоев
      layers = OrderedDict()
      for i in range(len(neur_counts) - 1):
         layers[str(i)] = (
            nn.Linear(
               neur_counts[i],
               neur_counts[i+1]
            )
         )
         layers[str(i)].weight = \
         nn.Parameter(
            torch.rand(
               layers[str(i)].weight.size()
            )
         )
         layers[str(i)].bias = \
         nn.Parameter(
            torch.rand(
               layers[str(i)].bias.size()
            )
         )
      self.layers = nn.Sequential(layers)

   def forward(self, x):
      for layer in self.layers[:-1]:
         x = F.relu(layer(x))

      x = torch.sigmoid(self.layers[-1](x))
      return x
```

Листинг К.1 – Класс описывающий модель использованную в работе

Примечания

1. Источник: собственная разработка на основе [17 с. 47],
2. Для корректной работы функций предварительно необходимо установить библиотеку numpy.

```
from copy import deepcopy

import numpy as np

class model_trainer():
    '''Класс реализует алгоритм обучения сети'''
    def __init__(
        self, model, optimizer,
        loss_fn, lr_scheduler = None
    ):
        # inputs:
        # model - модель которая подлежит обучению
        # optimizer - оптимизатор, который педполагается использовать
        # loss_fn - функция потерь
        # lr_shceduler - планировщих learning rate
        self.model = model
        self.optimizer = optimizer
        self.loss_fn = loss_fn

        self.train_loss = np.array([])
        self.test_loss = np.array([])

        # тут храним лучшую модель
        # из полученных если ошиентироваться
        # на ошибку на тестовых данных
        self.best_model = deepcopy(model)
        self.best_epoch = 0

        self.lr_scheduler = lr_scheduler


    def append_loses(self, train_loader, test_loader):
        '''Добавление ошибок рассчитанных
        на тренировочном и тестовом загрузчиках'''
        self.train_loss = np.append(
            self.train_loss,
            get_loss_value(
                self.loss_fn,
                self.model,
                train_loader
            )
        )
        self.test_loss = np.append(
            self.test_loss,
            get_loss_value(
                self.loss_fn,
                self.model,
                test_loader
            )
        )
```

```
def fit(self, train_loader, test_loader,
        epochs = 20, check_epoch = 1):
    '"Провести тренировку модели"'
    # inputs:
    # train_loader - загрузчик тренировочных данных
    # test_loader - загрузчик тестовых данных
    # epochs - число эпох для обучения алгоритма
    # check_epoces - число эпох после чего алгорим
    #               может быть оставлен и ведется
    #               регистрация лучшей модели

    # получаем начальную ошибку до
    # какого либо смещения весов
    self.append_loses(train_loader, test_loader)

    for epoch in range(epochs):

        self.model.train()
        for batch in train_loader:
            self.optimizer.zero_grad()
            inputs, targets = batch
            output = self.model(inputs)
            loss = self.loss_fn(output, targets)
            loss.backward()
            self.optimizer.step()
        if self.lr_scheduler:
            self.lr_scheduler.step()

        # вычисление ошибок на тестовой и
        # обучающих выборках на каждой эпохе
        self.append_loses(train_loader, test_loader)


        # работа с критериями остановки и
        # сохранения модели
        if epoch > check_epoch:
            # проверяем критерий остановки
            if self.test_loss[-1] > self.test_loss[-(check_epoch-1)]:
                return

            # в том случае, если последняя полученная
            # ошибка на тестовых данных наименьшая
            # то нужно запомнить модель как наилучшую
            if self.test_loss[-1] == np.min(self.test_loss):
                self.best_model = deepcopy(self.model)
                self.best_epoch = epoch
```

Листинг К.2 – Класс описывающий алгоритм обучения модели

Примечания:

1. Источник: собственная разработка на основе [17 с.47],
2. Для корректной работы функций предварительно необходимо установить библиотеку numpy.

```
from torch.utils.data import Dataset

class My_data_set(Dataset):
    '''Набор данных'''
    def __init__(self, X, Y):
        self.X = X
        self.Y = Y

    def __len__(self):
        return self.X.shape[0]

    def __getitem__(self, idx):
        return [self.X[idx,:], self.Y[idx, :]]
```

Листинг К.3 – Класс набора данных для создания загручиков

Примечания:

1. Источник: собственная разработка,
2. Для корректной работы функций предварительно необходимо установить библиотеку pytorch.

```
train_data = My_data_set(
    torch.tensor(X_train.astype('float32')),
    torch.tensor(y_train.astype('float32'))
)
train_data_loader =\
torch.utils.data.DataLoader(
    train_data, batch_size=500
)


test_data = My_data_set(
    torch.tensor(X_test.astype('float32')),
    torch.tensor(y_test.astype('float32'))
)
test_data_loader =\
torch.utils.data.DataLoader(
    test_data, batch_size=500
)

# создание модели
torch.manual_seed(0)
model_1lay = ResultNet([X.shape[1], X.shape[1]])

# оптимизатор
optimizer = optim.Adam(
    model_1lay.parameters(),
    lr = 0.01
)

lay1_trainer = model_trainer(
    model_1lay, optimizer, loss_fn
)

lay1_trainer.fit(
```

```
    train_data_loader,
    test_data_loader
)
```

Листинг К.4 – Первый запуск модели на обучение
Примечание – Источник: собственная разработка.