**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ)»**

**Журнал практики**

Студента Сорокина Дениса Михайловича

Факультет №8 «Информационные технологии и прикладная математика»

Кафедра 804 «Теория вероятностей и компьютерное моделирование»

Учебная группа 80-404-15Б

Направление подготовки (специальность) 01.03.02

*(шифр)*

Прикладная математика и информатика

*(название направления, специальности)*

Вид практики *преддипломная*

*(учебной, производственной, преддипломной или другой вид практики)*

Руководитель практики от МАИ

Соболь В.Р.

*(фамилия, имя, отчество) (подпись)*

Сорокин Д.М. / / “ 10 ” мая 2019 г.

*(подпись студента) (дата)*

Москва 2019

**1. Место и сроки проведения практики**

*Сроки проведения практики:*

*-дата начала практики* 09.02.2019

*-дата окончания практики* 10.05.2019

*Наименование предприятия*  Московский авиационный институт (национальный исследовательский университет)

*Название структурного подразделения (отдел, лаборатория)*  кафедра 804

**2. Инструктаж по технике безопасности**

/ Соболь В.Р. / “ 09 ” февраля 2019 г.

*(подпись проводившего) (дата проведения)*

**3. Индивидуальное задание студенту**

На наборе данных о финансовых транзакциях сделать классификатор мошеннических транзакций на основе модели «Случайный лес». Построить граф между участниками переводов, отметить тех, кто был задействован в мошеннических транзакциях. На графе посчитать такие параметры, как дистанция от текущего объекта до ближайшего мошенника, количество мошенников в круге радиуса 2-3, включить эти параметры в модель и проверить значимость.

**4. План выполнения индивидуального задания**

9 февраля – получение задания

10 февраля – 17 февраля – поиск и анализ литературных источников по тематике практики

18 февраля – 18 марта – Предварительный анализ данных. Разработка алгоритмов для построе-

ния и анализа графов. Написание программы на языке C#.

19 марта – 19 апреля – выбор алгоритмов классификации данных, написание программы на

языке Python. Анализ результатов

20 апреля – 30 апреля – оформление отчета по преддипломной практике

1 мая – 10 мая – подготовка к защите преддипломной практики

*Руководитель практики от МАИ:* Соболь В.Р./ /

*Руководитель от предприятия:* Соболь В.Р./ /

Сорокин Д.М. / / “ 09 ” февраля 2019 г.

*(подпись студента) (дата)*

**5. Отзыв руководителя практики от МАИ**

Практическое задание выполнено в полном объеме. В ходе выполнения задания студент продемонстрировала знание современных IT-технологий и математической статистики. Работа выполнена в значительной степени самостоятельно. Материалы, изложенные в отчёте студента, полностью соответствуют индивидуальному заданию. Полученный результат составляет основу дипломной работы. Оценка - отлично.

*Руководитель от МАИ:* Соболь В.Р. / /

*(фамилия, имя, отчество) (подпись)*

« 10 » мая 2019 г.

М.П. (печать)

**Отзыв руководителя практики от предприятия**

*Руководитель от предприятия:*  / /

*(фамилия, имя, отчество) (подпись)*

« 10 » мая 2019 г.

М.П. (печать)

**6. Отчет студента о практике**

Каждый день в мире совершаются миллионы финансовых транзакций. Хоть деньги и частично утратили свой физический, ощущаемый облик, мошенники, желающие их похитить, остались.

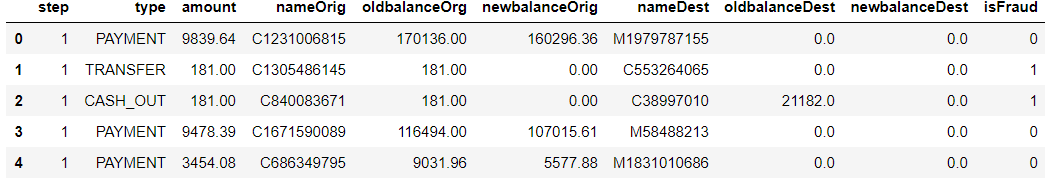
В связи с огромным количеством совершаемых транзакций, человеку будет практически нереально анализировать весь этот поток. На помощь ему приходят вычислительные машины, которые в последние годы стремительно развиваются и способны выполнять невероятные по объему и скорости вычисления.

В связи с этим набирает закономерную популярность такая наука как машинное обучение. С помощью предлагаемых ею методов возможно предсказыватьрезультат тех или иных задач.

В данной работе я постараюсь обучить модель, способную отличать мошеннические транзакции от немошеннических.

По соображениям конфиденциальности личных данных, в открытом доступе не имеется наборов реальных финансовых транзакций. Поэтому был выбран набор, созданный синтетически на основе реальных.

Вот его часть:



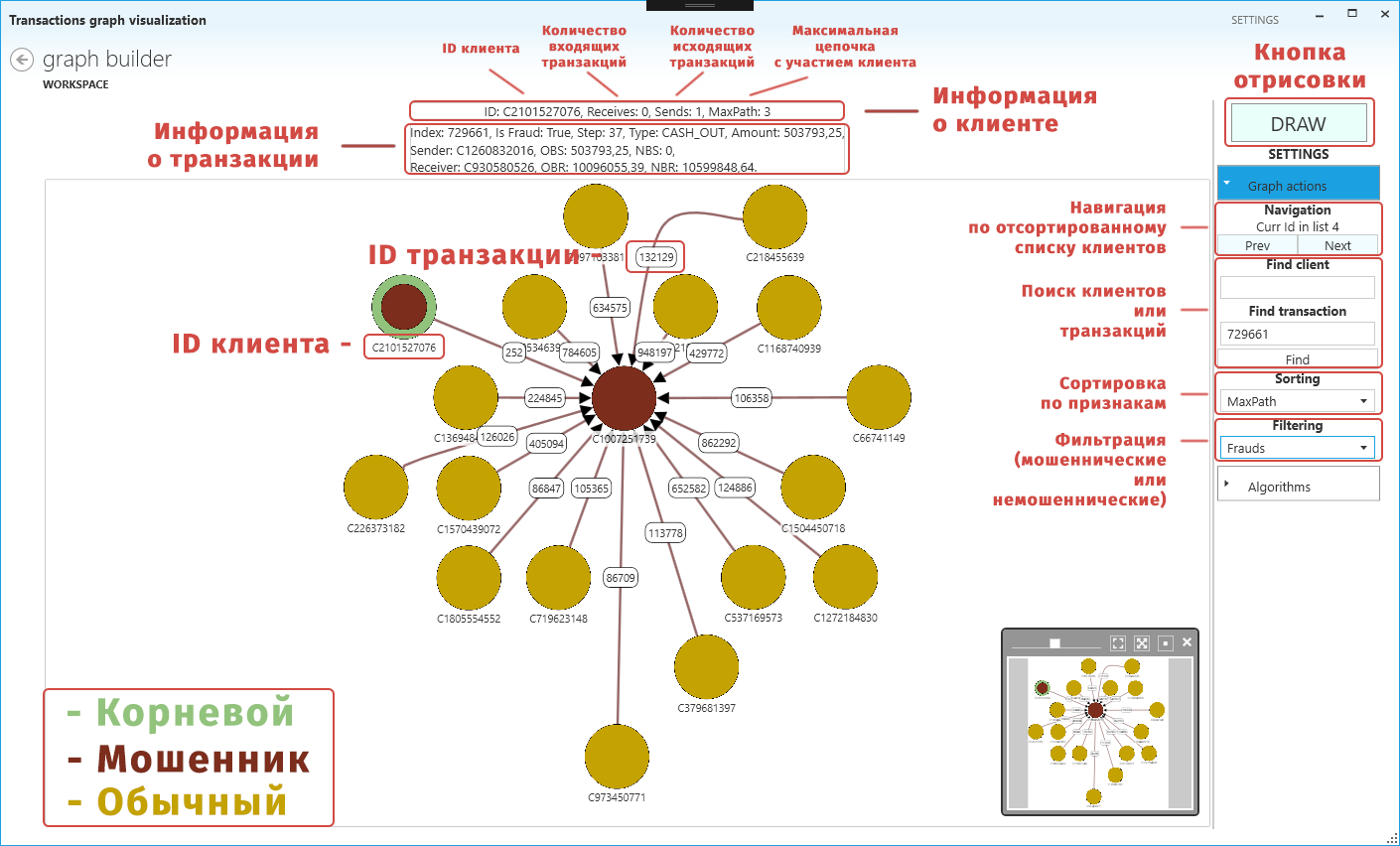
Он симулирует реальные мобильные переводы через некий африканский сервис в течение месяца.

Описание признаков

* step - аналог времени. 1 step = 1 час (всего 744 = 30 дней)
* type - тип транзакции (CASH-IN, CASH-OUT, DEBIT, PAYMENT, TRANSFER)
* amount - сумма перевода
* nameOrig - ID пользователя-отправителя
* oldbalanceOrg - баланс отправителя до транзакции
* newbalanceOrig - баланс отправителя после транзакции
* nameDest - ID пользователя-получателя
* oldbalanceDest - баланс получателя до транзакции. ID, начинающийся с буквы М - Merchant (магазин). В этом случае информация отсутствует
* newbalanceDest - баланс получателя после транзакции
* isFraud - пометка о мошеннической транзакции

1. Визуализация с помощью графов

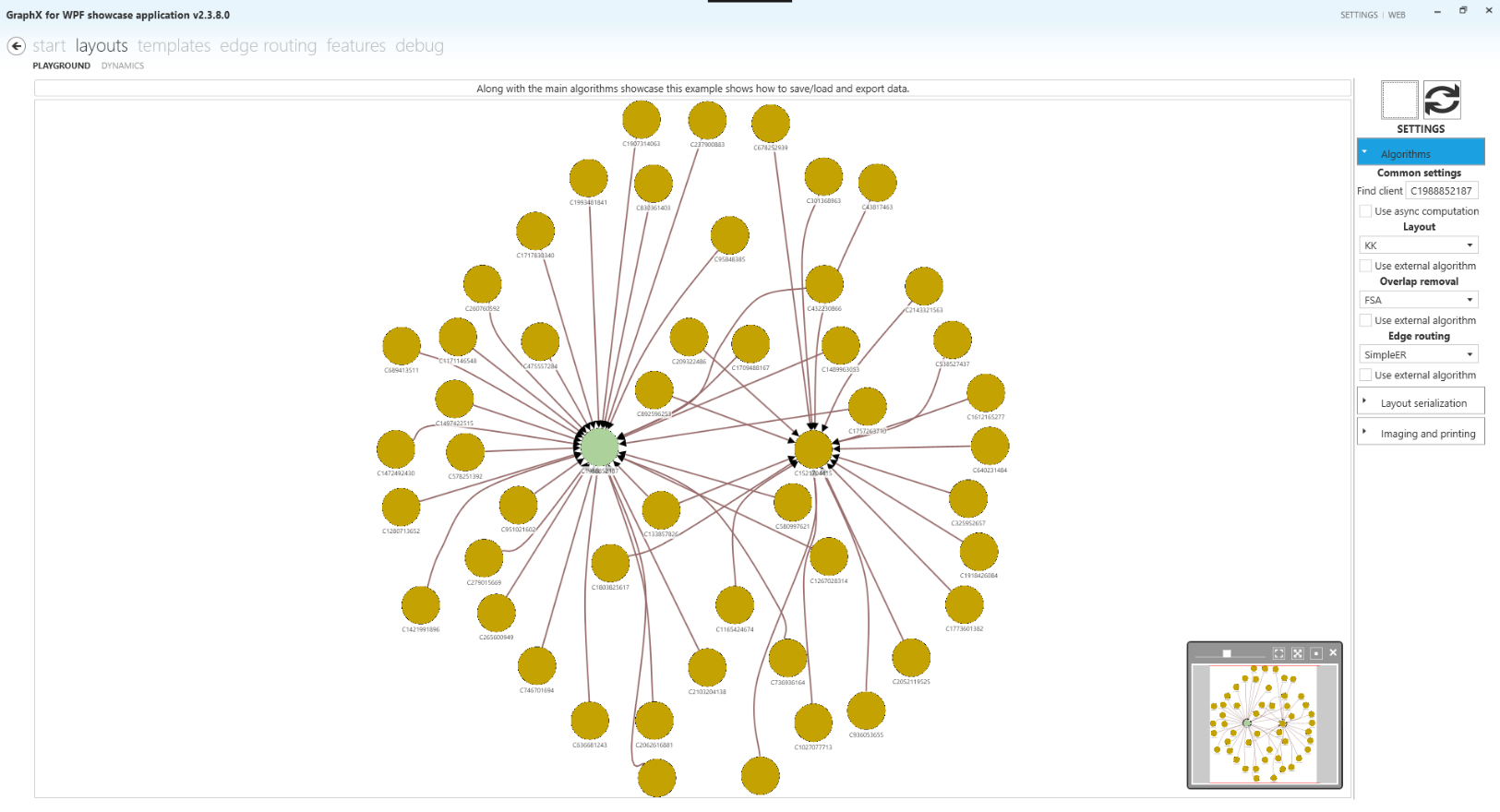
Для визуального анализа была написана программа, строящая и рисующая графы.

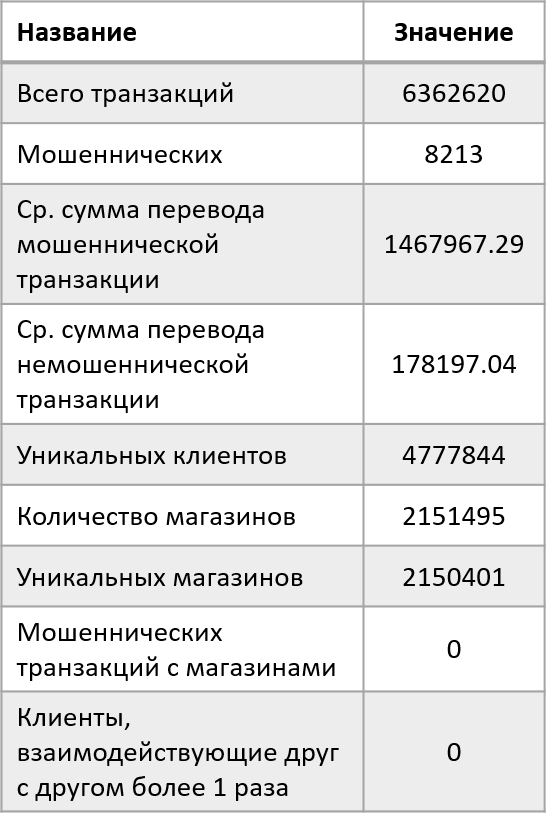


В процессе анализа стало ясно, что граф сильно разрежен, т.е. из шести с половиной миллионов транзакций самая длинная цепочка переводов всего лишь 11. Это означает, что эти транзакции представляют собой очень много маленьких отдельных графов. Поэтому поиск путей до мошенников чаще всего ничего не даст.

Типичная мошенническая транзакция выглядит так: мошенник – получатель, к нему стекаются многие транзакции, одна из которых – мошенническая.

Вообще довольно распространенная картина, когда к одному получателю стекается куча транзакций. Причем это клиент, а не магазин.

1. Предварительный анализ данных



Исходя из этих данных можно сделать выводы

1. Не мошеннических транзакций очень много, можно сократить в разы, чтобы модель обучалась адекватнее.
2. Количество уникальных клиентов опять же доказывает сильную разреженность этого набора.
3. Средняя сумма мошеннической транзакции на порядок превышает сумму не мошеннической. На основе этого можно будет делать предположения при классификации.
4. С магазином не производится мошеннических транзакций.
5. Нет клиентов, которые контактировали более одного раза. Не очень похоже на реальную ситуацию.
6. Мошеннические транзакции происходят в рамках двух типов.
7. Исследование признаков

Для того, чтобы модель лучше обучалась, можно предоставить ей скрытую информацию, которую можно извлечь из набора данных.

1. Перевести шаги в часы. Возможно мошеннические транзакции чаще совершаются в определенное время суток.
2. Указать выступал ли данный клиент в роли отправителя или получателя ранее. Т.к. было видно, что мошенники в основном получатели, это поможет при классификации.
3. Указать является ли магазином принимающая сторона. С магазинами не совершались мошеннические транзакции, так что можно будет сразу классифицировать как не мошенническая.
4. Были ли клиенты обрабатываемой транзакции ранее замечены в мошенничестве.
5. Время, прошедшее с предыдущей транзакции в качестве получателя/отправителя. Возможно мошеннические транзакции совершаются с определенной периодичностью или наоборот очень редкие на общем фоне.
6. Остается ли хоть что-нибудь на балансе у отправителя. Если нет, то вероятнее всего его обчищают.
7. Для некоторых моделей будет лучше, если разбить типы транзакций на бинарные признаки.

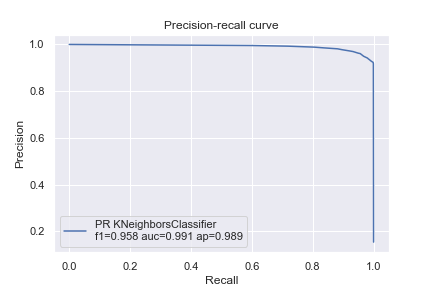
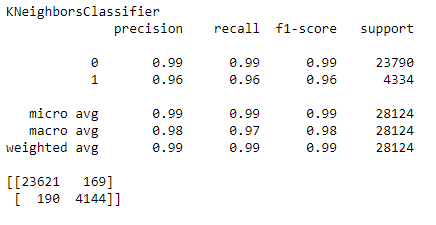
Также некоторые признаки можно убрать: они либо сильно коррелируют с имеющимися, либо бесполезны после проведения исследования. Можно отбросить:

1. Шаг, т.к. его заменили на часы
2. ID клиентов. Для работы модели они бесполезны.
3. Новый баланс отправителя. Сильно коррелирует с суммой перевода и старым балансом, в случае аномалии была введена переменная, помечающая нулевой остаток на счету.
4. Старый и новый баланс получателя. Ничего не дает о переводе, есть сумма.

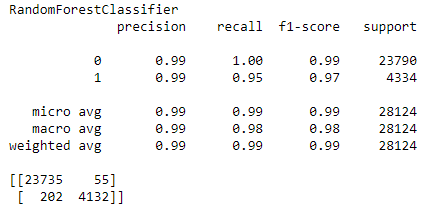
Далее будем экспериментировать и выбирать наиболее подходящую модель.

1. Эксперименты на моделях

Мною были выбраны 3 модели:

1. Случайный лес

Представляет собой ансамбль из решающих деревьев, решение принимается голосованием.

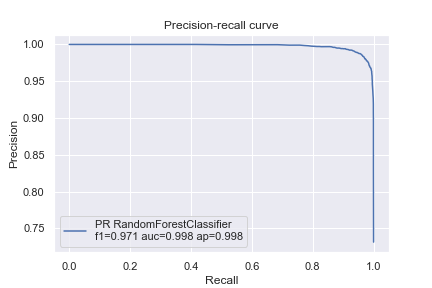


Две последние строчки – матрица ошибок

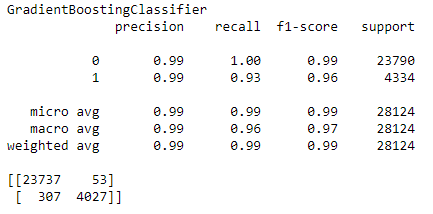
|  |  |
| --- | --- |
| True negative | False positive |
| False negative | True positive |

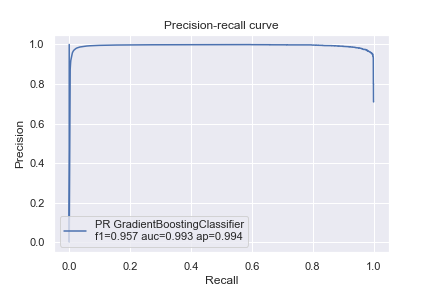
Выше указаны метрики, позволяющие оценить успешность предсказаний

* 
* 
* Гармоническое среднее для precision и recall называется F-мерой 
* Средняя точность (AP на графике) суммирует средневзвешенное значение точности, достигнутой на каждом пороге, с увеличением отзыва от предыдущего порога .

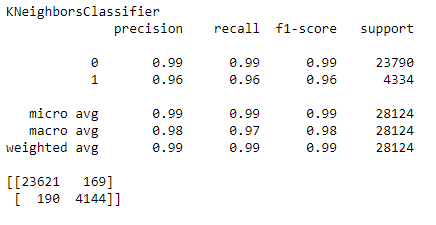
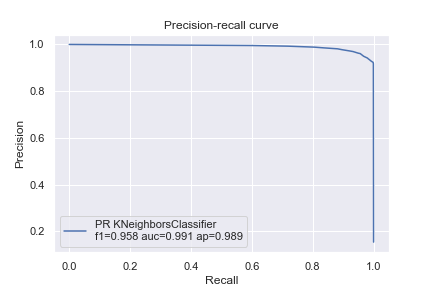


1. Градиентный бустинг





3. k-Ближайших соседей



Как видно из метрик лучше всего справился случайный лес.

Однако такие замечательные показатели довольно настораживают. Полученные результаты близки к идеальным. Скорее всего так получилось из-за того, что набор данных создан искусственно и существуют зависимости, которые позволяют достигнуть такой точности.

**Список литературы**

1. Ф.Харари ТЕОРИЯ ГРАФОВ М.: Мир, 1973, 300 стр.
2. Глубокое обучение, [Ян Гудфеллоу](https://www.google.ru/search?tbo=p&tbm=bks&q=inauthor:%22%D0%AF%D0%BD+%D0%93%D1%83%D0%B4%D1%84%D0%B5%D0%BB%D0%BB%D0%BE%D1%83%22), [Иошуа Бенджио](https://www.google.ru/search?tbo=p&tbm=bks&q=inauthor:%22%D0%98%D0%BE%D1%88%D1%83%D0%B0+%D0%91%D0%B5%D0%BD%D0%B4%D0%B6%D0%B8%D0%BE%22), [Аарон Курвилль](https://www.google.ru/search?tbo=p&tbm=bks&q=inauthor:%22%D0%90%D0%B0%D1%80%D0%BE%D0%BD+%D0%9A%D1%83%D1%80%D0%B2%D0%B8%D0%BB%D0%BB%D1%8C%22)
3. Python Machine Learning Book by Sebastian Raschka
4. Загоруйко Н. Г. Прикладные методы анализа данных и знаний.
5. Лекции по логическим алгоритмам классификации, К.В. Воронцов, 2010 г

**Приложение**

Код программ

Python

import numpy as np

import random

from pandas import read\_csv as read

import pandas as pd

import seaborn as sns; sns.set(color\_codes=True)

import matplotlib.pyplot as plt

path = "newFilt.csv"

data = read(path, delimiter=",")

print(data.corr())

# print(data.head())

data.drop('isFlaggedFraud', axis=1, inplace=True)

data.drop('nameOrig', axis=1, inplace=True)

data.drop('nameDest', axis=1, inplace=True)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

data['type'] = le.fit\_transform(data['type'])

# data = pd.get\_dummies(data)

print(data.head())

# print(data.head())

X = data.loc[:, data.columns != 'isFraud'].values

y = data.loc[:, 'isFraud'].values

y=y.astype('int') # не совсем понял, почему https://stackoverflow.com/questions/45346550/valueerror-unknown-label-type-unknown

# from sklearn import preprocessing

# # normalize the data attributes

# normalized\_X = preprocessing.normalize(X)

# # standardize the data attributes

# standardized\_X = preprocessing.scale(X)

from sklearn.model\_selection import train\_test\_split as train

X\_train, X\_test, y\_train, y\_test = train(X, y, shuffle=False)

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn import metrics

models = []

models.append(DecisionTreeClassifier())

models.append(RandomForestClassifier(n\_estimators=110, n\_jobs=-1))

models.append(GradientBoostingClassifier(max\_depth=4))

models.append(KNeighborsClassifier(n\_neighbors=20))

for model in models:

model.fit(X\_train, y\_train)

expected = y\_test

predicted = model.predict(X\_test)

pobas = model.predict\_proba(X\_test)

preds = pobas[:, 1]

report = metrics.classification\_report(expected, predicted)

conf\_matrix = metrics.confusion\_matrix(expected, predicted)

print(type(model).\_\_name\_\_)

print(report)

print(conf\_matrix)

print(metrics.roc\_auc\_score(expected, preds))

# print(roc\_auc)

# df = pd.DataFrame()

# df['Exp'] = expected

# df['pre'] = predicted

# df.to\_csv('ep.csv', index=False)

plt.savefig("roc.png")

С#

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Windows;

namespace DiplomaHelp

{

class GraphDataManager

{

public HashSet<string> IsVisited = new HashSet<string>();

public List<string> StringsFile = new List<string>();

public List<Transaction> TransactionsList = new List<Transaction>();

public Dictionary<string, int> StringTrDict = new Dictionary<string, int>();

public List<string> ClientsIdList = new List<string>();

public Dictionary<string, Client> ClientStringClassDict = new Dictionary<string, Client>();

public Dictionary<Client, List<Client>> ClientsGraph = new Dictionary<Client, List<Client>>();

public Dictionary<string, int> MerchantsDict = new Dictionary<string, int>();

public List<Client> CurrWorkingList = new List<Client>();

public int CurrentRootClientInd = 0;

public bool IsSortingChanged = false;

public bool IsFilterChanged = false;

public GraphDataManager(string diplomaDataCsv)

{

StringsFile = File.ReadAllLines(diplomaDataCsv).ToList();

StringsFile.RemoveAt(0); // удаляется строка с описанием столбцов

//var strCount = StringsFile.Count;

//var strCount = StringsFile.Count / 2;

var strCount = 100000;

for (int i = 0; i < strCount; i++)

{

var el = StringsFile[i];

TransactionsList.Add(new Transaction(el));

StringTrDict.Add(TransactionsList[i].NameOrig + TransactionsList[i].NameDest, i);

var sender = TransactionsList[i].NameOrig;

var receiver = TransactionsList[i].NameDest;

if (receiver.Contains("M"))

{

if (MerchantsDict.ContainsKey(receiver))

MerchantsDict[receiver]++;

else

MerchantsDict.Add(receiver, 0);

continue;

}

ClientsIdList.Add(sender);

ClientsIdList.Add(receiver);

}

ClientsIdList = ClientsIdList.Distinct().ToList();

foreach (var el in ClientsIdList)

{

var tmpClient = new Client(el);

ClientsGraph.Add(tmpClient, new List<Client>());

ClientStringClassDict[el] = tmpClient;

}

for (var i = 0; i < strCount; i++)

{

var tr = TransactionsList[i];

if (tr.NameDest.Contains("M"))

continue;

var sender = ClientStringClassDict[tr.NameOrig];

var receiver = ClientStringClassDict[tr.NameDest];

if (tr.isFraud)

{

sender.IsFraud = true;

receiver.IsFraud = true;

}

sender.AddToSender(tr);

receiver.AddToReceiver(tr);

ClientsGraph[sender].Add(receiver);

ClientsGraph[receiver].Add(sender);

}

foreach (var pair in ClientsGraph)

{

pair.Key.MaxPath = DFS(pair.Key, pair.Value, ClientsGraph, pair.Key.MaxPath);

}

FilterByCriteria(FilterTypes.All);

SortByCriteria(SortTypes.MaxPath);

MessageBox.Show("Data downloading completed");

}

public void SortByCriteria(SortTypes criteria)

{

if (criteria == SortTypes.MaxPath)

CurrWorkingList = CurrWorkingList.OrderBy(c => -c.MaxPath).ToList();

if (criteria == SortTypes.Receives)

CurrWorkingList = CurrWorkingList.OrderBy(c => -c.Receiver.Count).ToList();

if (criteria == SortTypes.Sends)

CurrWorkingList = CurrWorkingList.OrderBy(c => -c.Sender.Count).ToList();

var tmpstr = new List<string>();

foreach (var el in CurrWorkingList)

{

tmpstr.Add(el.ToString());

}

File.WriteAllLines("path.txt", tmpstr);

}

public void FilterByCriteria(FilterTypes criteria)

{

if (criteria == FilterTypes.All)

CurrWorkingList = ClientsGraph.Keys.ToList();

if (criteria == FilterTypes.Frauds)

CurrWorkingList = CurrWorkingList.FindAll(el => el.IsFraud);

}

private int DFS(Client client, List<Client> connectClients, Dictionary<Client, List<Client>> clientsGraph, int path)

{

var currId = client.Id;

if (IsVisited.Contains(currId))

return path;

if (connectClients == null || connectClients.Count == 0)

return path + 1;

IsVisited.Add(currId);

++path;

var lenghtList = new List<int>();

foreach (var cl in connectClients)

{

cl.MaxPath = DFS(cl, clientsGraph[cl], clientsGraph, cl.MaxPath);

lenghtList.Add(cl.MaxPath);

}

return path + lenghtList.Max();

}

}

}