ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

**Институт кибербезопасности и защиты информации**

## Отчет о прохождении производственной практики (научно-исследовательская работа)

Кулеева Анна Григорьевна

*(Ф.И.О. обучающегося)*

4 курс, 4851003/90801

*(номер курса обучения и учебной группы)*

10.05.03 Информационная безопасность автоматизированных систем

*(направление подготовки (код и наименование)*

## Место прохождения практики: СПбПУ, Институт кибербезопасности и защиты

*(указывается наименование профильной организации или наименование структурного подразделения*

информации, г. Санкт-Петербург, Политехническая ул., д.29

*ФГАОУ ВО «СПбПУ», фактический адрес)*

## Сроки практики: 02.02.2023–24.05.2023 Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»:

Платонов В.В., к.т.н., доцент

*(Ф.И.О., уч.степень, должность)*

**Оценка:**

Руководитель практической подготовки

от ФГАОУ ВО «СПбПУ»: В.В. Платонов

Обучающийся: А.Г. Кулеева

Дата: 23.06.2023

**Содержание**

1. Цель и задачи научно-исследовательской работы 3
2. Ход работы 3
   1. [Метод главных компонент](#_Toc506483745) 3
   2. Построение нейронной сети и подбор параметров 5
   3. Оценка разработанной модели 6
3. [Выводы](#_Toc506483745) 7
4. Список использованных источников [8](#_Toc506483746)

Приложение А9

1. **Цели и задачи практики**

На период практики было получено следующее задание:

1. Проанализировать набор данных, выбрать главные компоненте для обучения НС.
2. Разработать программный прототип персептрона и подобрать для него параметры, дающие лучший результат при обучении.
3. Обучить НС и проанализировать полученные результаты.
4. **Ход работы**

**2.1.** **Метод главных компонент**

Метод главных компонент является одним из алгоритмов сокращения размерности обрабатываемых данных с наименьшей потерей информации в процессе машинного обучения. Суть метода заключается в следующем. Строится матрица, где каждая строка ― выборка, то есть числовая характеристика некоторой компоненты. Далее строится ковариационная матрица. Она является обобщением дисперсии на случай многомерных случайных величин и описывает разброс случайной величины так же, как и дисперсия. Диагональные элементы ковариационной матрицы показывают дисперсии по изначальному базису, а ее собственные значения ― по новому (по главным компонентам). Далее необходимо найти собственный вектор матрицы. При перемножении собственного вектора на матрицу ковариации будет получен новый вектор ― проекция, содержащий в себе значения главной компоненты.

Был использован набор данных CSE-CIC-IDS2018 [6]. В нем используется понятие профилей для систематического создания наборов данных, которые будут содержать подробные описания вторжений и абстрактные модели распространения для приложений, протоколов или сетевых объектов более низкого уровня. Эти профили могут использоваться для генерации событий в сети. Благодаря абстрактному характеру сгенерированных профилей их можно применять к широкому спектру сетевых протоколов с различной топологией.

Из датасета были взяты записи и на основе них построены выборки размера 10000 для каждой компоненты. В языке программирования Python метод главных компонент уже реализован в библиотеке sklearn [3]. Необходимо задать, до какого числа компонент будет снижена размерность, и передать методу матрицу значений. В качестве возвращаемого значения будут получены собственные вектора с максимальными собственными числами.

Необходимо выделить столько главных компонент, чтобы они охватывали 99% информации. Формула для вычисления представлена ниже:

Для получения этих данных воспользуемся методом explained\_variance\_ratio\_ того же класса PCA библиотеки sklearn [4].

При помощи метода главных компонент были выбраны 5 основных параметров, которые содержат в себе 98% информации. Были выбраны следующие параметры (Рисунок ): продолжительность потока, среднее время между двумя пакетами, отправленными в обратном направлении, стандартное отклонение времени между двумя пакетами, отправленными в обратном направлении, минимальное время между двумя потоками, максимальное время простоя потока до того, как он стал активным. При чем параметр bw\_iat\_avg повторяется два раза, но имеет разные значения. В ходе поиска данной ошибки в сети Интернет было выяснено, что преобразование PCA не дает гарантии, что входные данные будут разными. Такая функция вносит наибольший вклад, поэтому получаются дублирующиеся компоненты [2]. Таким образом, данный дубликат игнорируется.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 — Главные компоненты

**2.2. Построение нейронной сети и подбор параметров**

Перед построением модели машинного обучения необходимо подготовить входные данные. Была создана выборка размером 20000. Проверим, сколько лейблов содержат 0, а сколько 1: выборка оказалась несбалансированной. Исправим это методом upsampling, то есть, методом дублирования тех компонент, которых меньше, пока отношение не будет примерно одинаковым [5]. На Рисунок видно отношение классов до и после преобразования.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок 2 — Балансировка выборки данных

В качестве модели машинного обучения был выбран многослойный персептрон [7, 8, 9]. С помощью фреймворка scikit-learn 1.2.2 был реализован персептрон [1], а именно модель MLP. Реализованный персептрон имеет следующую структуру: n=5 входных нейронов, 2 скрытых слоя, 1 выходной нейрон, который выдает либо 0 (атаки нет), либо 1 (атака есть). Рассматриваются различные комбинации нейронов скрытого слоя от 0 до 2n+1 на каждом слое. На каждом шаге вычисляются ошибка, точность и AUC. Веса инициализируются библиотекой самостоятельно исходя из количества нейронов, либо могут быть указаны вручную при необходимости. На Рисунок представлен график изменения значения ошибки при различном количестве нейронов в сети.

Оптимальный результат был получен при 11 нейронах на каждом слое. Однако стоит заметить, что значения могут меняться при каждом запуске алгоритма, поскольку в нем присутствует некоторый элемент случайности, а именно случайное разбиение всей выборки на обучающую и валидационную, а также случайное распределение весов.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 3 — Изменение значения ошибки при различном числе нейронов скрытого слоя

**2.3. Оценка разработанной модели**

Рассмотрим более подробно метрики для каждого класса при 11 нейронах в каждом скрытом слое. На Рисунок представлена матрица несоответствия, precision, recall, F-мера для каждого класса в отдельности и в обоих в совокупности, а также посчитан параметр AUC. На Рисунок представлен график ROC-кривой.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 — Метрики качества разработанной модели

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 5 — ROC

1. **Выводы**

В ходе выполнения данной работы был реализован однослойный персептрон с 5 нейронами входного слоя, 11 нейронами на каждом из 2 скрытых слоев и 1 выходным нейроном. В качестве функции активации использовалась ReLU, в качестве функции ошибки — бинарная кросс-энтропия, в качестве оптимизатора — Адам.

При помощи метода главных компонент были выбраны 5 основных параметров, которые содержат в себе 98% информации. Были выбраны следующие параметры: продолжительность потока, среднее время между двумя пакетами, отправленными в обратном направлении, стандартное отклонение времени между двумя пакетами, отправленными в обратном направлении, минимальное время между двумя потоками, максимальное время простоя потока до того, как он стал активным.

Для улучшения качества модели она была обучена при различных комбинациях параметров. Оптимизатор Адам показал лучший результат по сравнению с алгоритмом SGD. Однако разница была не сильно значительной.

В итоге показатель AUC составил лишь 0,77. При проверке разработанной модели на другом наборе данных она показала результат свыше 0,95. Из чего можно сделать вид, что конкретно модель MLP может быть не такой эффективной в применении к данному набору данных.

**Список использованных источников**

1. sklearn.neural\_network.MLPClassifier [Электронный ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html> . – (дата обращения: 06.05.2023).
2. Why PCA output some components duplicately? [Электронный ресурс]. URL: <https://stackoverflow.com/questions/67769996/why-pca-output-some-components-duplicately>. – (дата обращения: 26.06.2021).
3. Как работает метод главных компонент (PCA) на простом примере [Электронный ресурс]. URL: <https://habr.com/ru/post/304214/> . – (дата обращения: 12.06.2022).
4. [sklearn.decomposition](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition).PCA [Электронный ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA.get_feature_names_out>. – (дата обращения: 26.06.2022).
5. Как победить несбалансированность датасета: метод upsampling data [Электронный ресурс]. URL: <https://habr.com/ru/articles/568266/>. – (дата обращения: 21.05.2023).
6. Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, «[Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization](https://www.semanticscholar.org/paper/Toward-Generating-a-New-Intrusion-Detection-Dataset-Sharafaldin-Lashkari/a27089efabc5f4abd5ddf2be2a409bff41f31199)», 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018
7. Д. Рутковская, М. Пилиньский, Л. Рутковский Нейронные сети, генетические алгоритмы и нечеткие системы //Горячая линия – Телеком. – 2006. – №. 103. – С. 18-25.
8. Фрэнк Розенблатт [Принципы нейродинамики: перцептроны и теория механизмов мозга](http://www.raai.org/library/books/other/ros_principles2.djvu) //М.: «Мир», 1965. – С. 43.
9. Ф. Вассерман Нейрокомпьютерная техника: Теория и практика. // М.: «Мир», 1992.

# Приложение

import pandas as pd  
from math import sqrt  
from sklearn.neural\_network import MLPClassifier  
from sklearn.preprocessing import Normalizer  
from sklearn.model\_selection import train\_test\_split, GridSearchCV, RandomizedSearchCV  
from sklearn.metrics import \*  
import matplotlib.pyplot as plt  
import warnings  
from sklearn.exceptions import DataConversionWarning  
warnings.filterwarnings(action='ignore', category=DataConversionWarning)  
  
def main():  
 # how much neurons we need?  
 Q1 = Q2 = 12  
  
 hidden = [11, 11]  
 loss = []  
 my\_dict = {'loss': 1, 'layer1': 0, 'layer2': 0}  
 X = pd.read\_csv('nir.csv', sep=',', usecols=[1, 2, 3, 4, 5])  
 Y = pd.read\_csv('nir.csv', sep=',', usecols=[6])  
 X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, train\_size=0.5, random\_state=42)  
 # test\_data = pd.read\_csv('validate.csv', sep=',')  
 X\_train = Normalizer().fit\_transform(X\_train)  
 X\_test = Normalizer().fit\_transform(X\_test)  
 # create the model  
  
 print("start training...")  
 clf = MLPClassifier(hidden\_layer\_sizes=hidden, activation="relu", solver='adam',  
 max\_iter=1000, shuffle=True)  
 # print("neurons in hidden layer ", i, " ", j)  
 clf.fit(X\_train, Y\_train)  
 print("start validating...")  
 y\_pred = clf.predict(X\_test)  
 loss\_i = zero\_one\_loss(Y\_test, y\_pred)  
  
 print("loss = ", loss\_i)  
 print("mean test accuracy = ", clf.score(X\_test, Y\_test)) # Return the mean accuracy on the given test data and labels.  
 print("confusion matrix:\n", confusion\_matrix(Y\_test, y\_pred))  
 print(classification\_report(Y\_test, y\_pred))  
  
 #roc-auc  
 print("AUC = ", roc\_auc\_score(Y\_test, y\_pred))  
 RocCurveDisplay.from\_predictions(Y\_test, y\_pred)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()