Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский ядерный университет
МИФИ
(НИЯУ МИФИ)

УДК 53-07

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Изучение нейронных сетей и использование их для решения задач классификации

Научный руководитель:
д.ф. -м.н. _____А. А. Соколов

Выполнил: _____Т. В. Махкамов

Москва 2021

СОДЕРЖАНИЕ

Цель работы

Введение в нейронные сети и машинное обучение, а также использование их для решения задач классификации на примере решения задачи «Титаник».

1.1 Модель искусственного нейрона

1.1.1 Биологический нейрон

Нейронная сеть - математическая модель, построенная по принципу организации и функционирования сетей нервных клеток живого организма.

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами. Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от нашей кожи, ушей и глаз к мозгу, процессы мышления и управления действиями – все это реализовано в живом организме как передача электрических импульсов между нейронами.

Рис. 1.1 Биологический (или естественный) нейрон

Биологический нейрон имеет ядро, а также отростки нервных волокон двух типов (рис. 1.1) – дендриты, по которым принимаются импульсы, и

единственный аксон, по которому нейрон может передавать импульс. Аксон контактирует с дендритами других нейронов через специальные образования – синапсы, которые влияют на силу передаваемого импульса.

Структура, состоящая из совокупности большого количества таких нейронов, получила название биологической (или естественной) нейронной сети.

1.1.2 Искусственный нейрон

Искусственный нейрон (далее – нейрон) является основой любой искусственной нейронной сети. Нейроны представляют собой относительно простые, однотипные элементы, имитирующие работу нейронов мозга. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены и заторможены. Искусственный нейрон, также как и его естественный прототип, имеет группу синапсов (входов), которые соединены с выходами других нейронов, а также аксон – выходную связь данного нейрона – откуда сигнал возбуждения или торможения поступает на синапсы других нейронов.

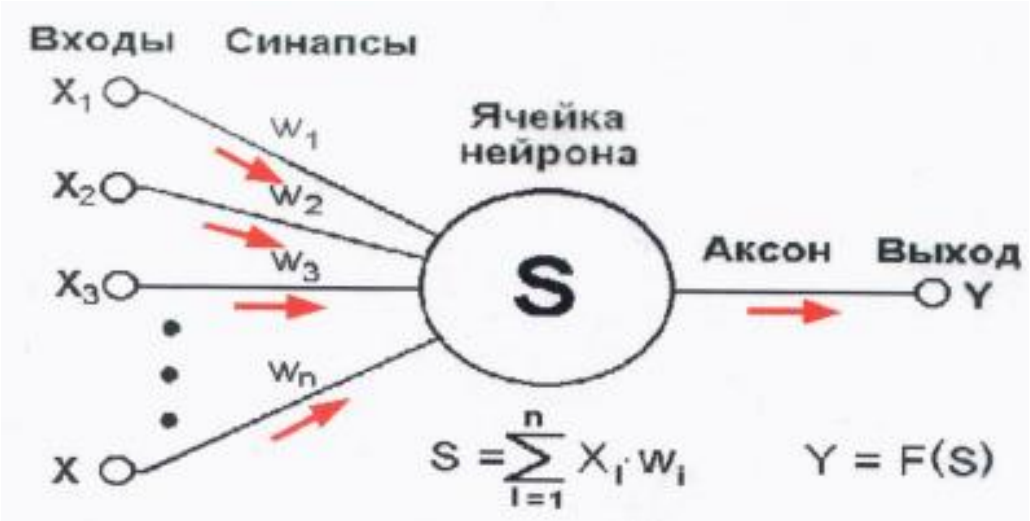


Рис. 1.2 Математическая модель нейрона

Каждый синапс характеризуется величиной *синаптической* связи или весом *w*, который по своему физическому смыслу эквивалентен электрической

проводимости. Текущее состояние нейрона определяется как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i w_i \ , \qquad (2.1)$$

где x – вход нейрона, а w – соответствующий этому входу вес.

1.1.3 Активационная функция

Выход нейрона есть функция его состояния, т.е.

$$y = f(s) \ . \qquad (2.2)$$

Нелинейная функция f(s) называется активационной, сжимающей функцией или функцией возбуждения нейрона. Основные разновидности активационных функций, применяемых в нейронных сетях, представлены на рис. 2.3.

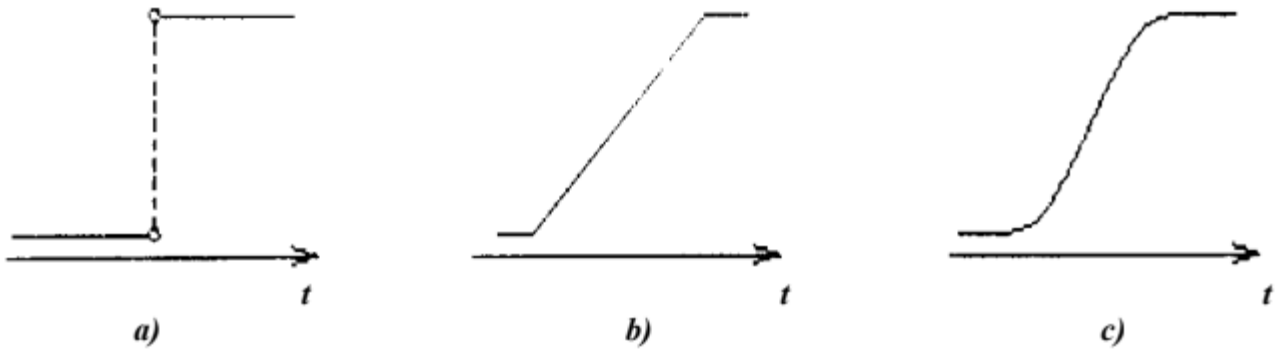


Рис. 1.3 Активационная функция

а) пороговая; b) полулинейная; c) сигмоидальная

В качестве активационной функции часто используется сигмоидальная (s-образная или логистическая) функция, показанная на рис. 1.3 ***c***. Эта функция математически выражается по формуле

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \ . \qquad (2.3)$$

При уменьшении a сигмоидальная функция становится более пологой, в

пределе при a=0 вырождаясь в горизонтальную линию на уровне 0,5; при увеличении a сигмоидальная функция приближается по внешнему виду к функции единичного скачка с порогом T в точке x=0. Из выражения для сигмоидальной функции видно, что выходное значение нейрона лежит в диапазоне [0,1].

1.1.4 Функция потерь

Функция потерь находится в центре нейронной сети. Она используется для расчета ошибки между реальными и полученными ответами. Наша глобальная цель — минимизировать эту ошибку. Таким образом, функция потерь эффективно приближает обучение нейронной сети к этой цели.

Функция потерь измеряет «насколько хороша» нейронная сеть в отношении данной обучающей выборки и ожидаемых ответов. Она также может зависеть от таких переменных, как веса и смещения.

Функция потерь одномерна и не является вектором, поскольку она оценивает, насколько хорошо нейронная сеть работает в целом.

Некоторые известные функции потерь:

- Квадратичная (среднеквадратичное отклонение);
- Кросс-энтропия;
- Экспоненциальная (AdaBoost);

Среднеквадратичное отклонение – самая простая функция потерь и наиболее часто используемая.

1.2 Обучение нейронных сетей

Функционирование нейронной сети, т. е. действия, которые она способна выполнять, зависит от величин синоптических связей. Поэтому, задавшись структурой нейронной сети, отвечающей определенной задаче, разработчик должен найти оптимальные значения для всех весовых коэффициентов *w*.

Этот этап называется обучением нейронной сети, и от того, насколько

качественно он будет выполнен, зависит способность сети решать во время эксплуатации поставленные перед ней проблемы. Важнейшими параметрами обучения являются: качество подбора весовых коэффициентов и время, которое необходимо затратить на обучение. Как правило, два этих параметра связаны между собой обратной зависимостью и их приходится выбирать на основе компромисса.

В настоящее время все алгоритмы обучения нейронных сетей можно разделить на два больших класса: с учителем и без учителя.

1.2.1 Обучение с учителем

Нейронной сети предъявляются значения как входных, так и выходных параметров, и она по некоторому внутреннему алгоритму подстраивает веса своих синаптических связей.

Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются представительской или обучающей выборкой. Обычно нейронная сеть обучается на некотором числе таких выборок. Предъявляется выходной вектор, вычисляется выход нейронной сети и сравнивается с соответствующим целевым вектором, разность (ошибка) с помощью обратной связи подается в нейронную сеть, и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

1.2.2 Обучение без учителя

Нейронной сети предъявляются только входные сигналы, а выходы сети формируются самостоятельно с учетом только входных и производных от них сигналов.

Несмотря на многочисленные прикладные достижения, обучение с учителем критиковалось за свою биологическую неправдоподобность. Трудно

вообразить обучающий механизм в естественном человеческом интеллекте, который бы сравнивал желаемые и действительные значения выходов, выполняя коррекцию с помощью обратной связи. Если допустить подобный механизм в человеческом мозге, то откуда тогда возникают желаемые выходы? Обучение без учителя является более правдоподобной моделью обучения в биологической системе.

Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса нейронной сети так, чтобы получались согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения, следовательно, выделяет статистические свойства обучающего множества и группирует сходные векторы в классы. Предъявление на вход вектора из данного класса даст определенный выходной вектор, но до обучения невозможно предсказать, какой выход будет производиться данным классом входных векторов.

2.1 Постановка задачи

Рассмотрим решение классификационной задачи с использованием аппарата нейронных сетей и других классификаторов на примере задачи “Титаник”. Это известная задача, которая рассматривается в рамках соревнования **Titanic: Machine Learning from Disaster** проходящего на сайте Kaggle. Датасет Титаник содержит реальные данные пассажиров корабля. Цель задачи — построить модель, которая лучшим образом сможет предсказать, остался ли произвольный пассажир в живых или нет, т.е. решить классификационную задачу. Для решения этой задачи использовался язык программирования Python.

Если доверять Википедии, то Титаник столкнулся с айсбергом в 11:40 вечера корабельного времени, когда подавляющее большинство пассажиров и корабельной команды находились в своих каютах. Соответственно,

расположение кают, возможно, имело влияние на вероятность выжить, т.к. пассажиры нижних палуб, во-первых, позднее узнали о столкновении и, соответственно, имели меньше времени добраться до верхней палубы. И, во-вторых, им, естественно, было дольше выбираться из помещений корабля. Ниже [Рис. 1.4] изображена схема Титаника с указанием палуб и помещений.

Титаник являлся британским кораблем, а согласно законам Британии на корабле должно было быть число шлюпок, соответствующее водоизмещению судна, а не пассажировместимости. Титаник формально соответствовал этим требованиям и имел 20 шлюпок (14 со вместимостью 65 человек, 2 — 40 человек, 4 — 47 человек), которые были рассчитаны на погрузку 1178 человек, всего же на Титанике было 2208 человек. Таким образом, зная, что шлюпок на всех не хватит, капитан Титаника Смит отдал, после столкновения с айсбергом, приказ брать на шлюпки только женщин и детей. Однако члены команды не всегда следовали ему.

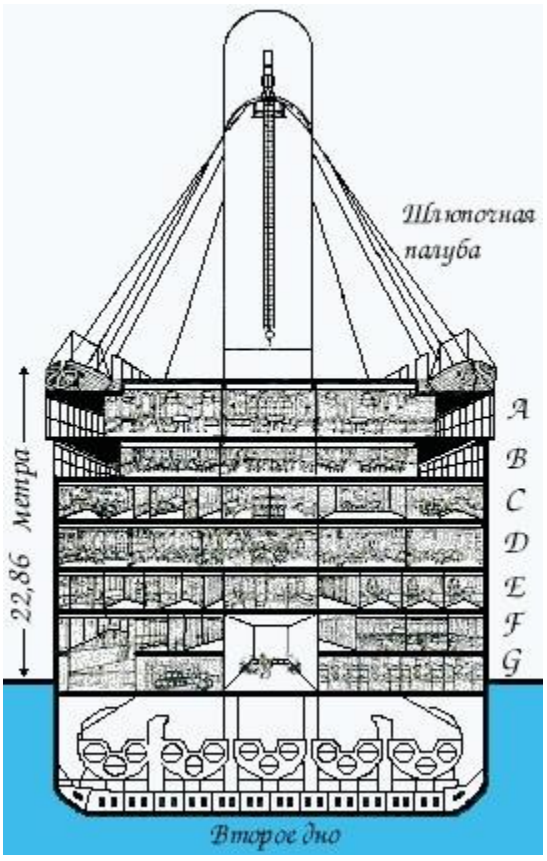


Рис. 1.4 Схема Титаника с указанием палуб и помещений

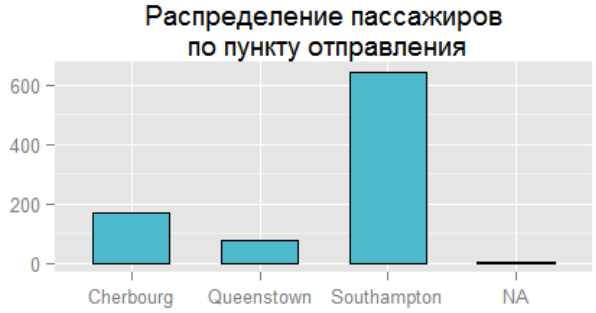
2.2 Изучение данных

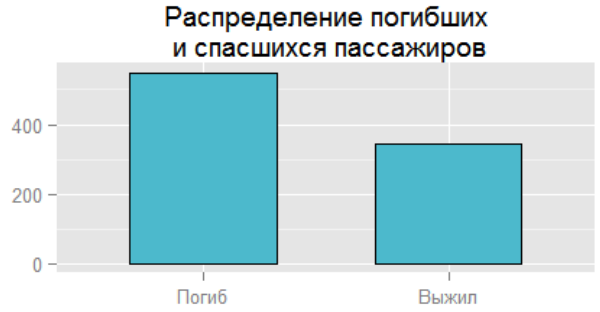
Прежде, чем приступить к решению классификационной задачи, необходимо изучить структуру используемых данных.

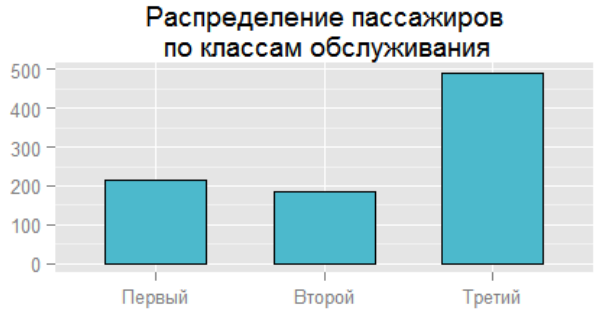
Kaggle предоставляет данные в виде двух файлов в формате csv:

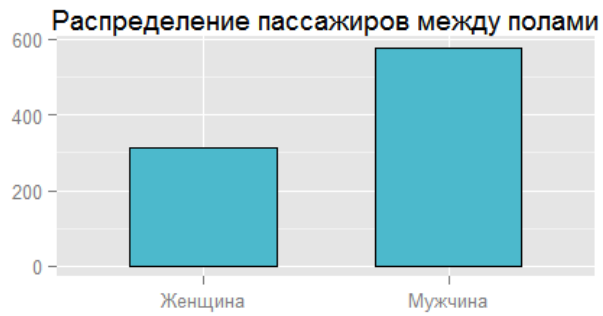
- train.csv (содержит выборку пассажиров с известным исходом, т.е. выжил или нет)
- test.csv (содержит другую выборку пассажиров без зависимой переменной)

Для анализа данных были построены графики зависимостей различных параметров пассажиров:









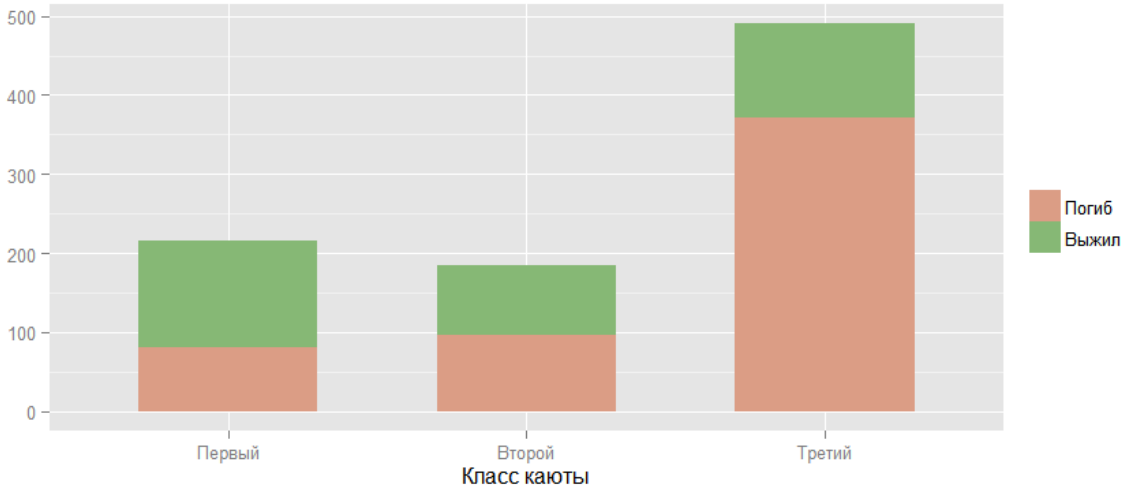
Уже можно делать первые выводы:

- больше пассажиров погибло, чем спаслось

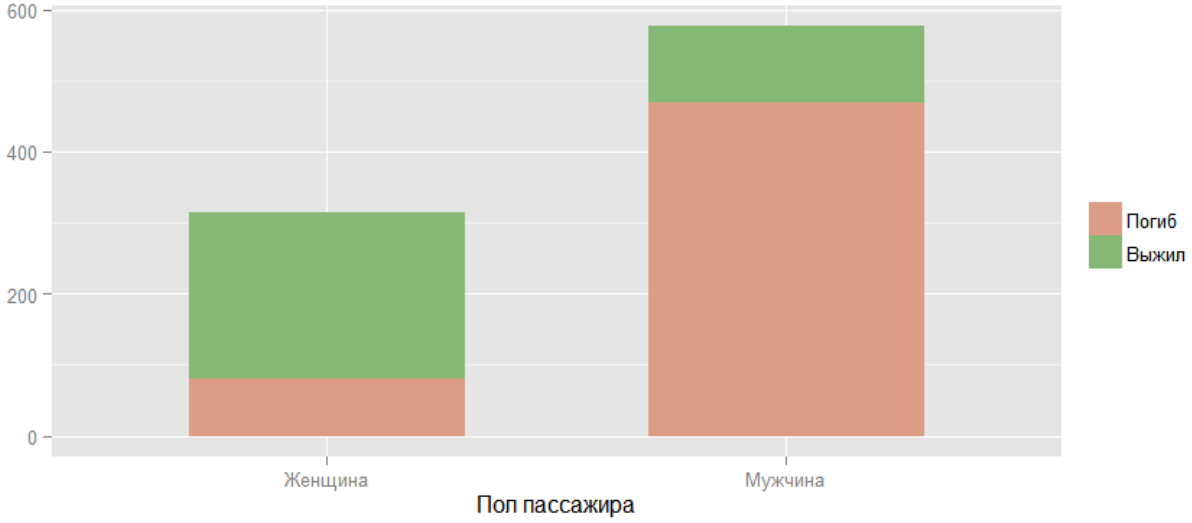
- подавляющее большинство пассажиров находилось в каютах третьего класса
- мужчин было больше чем женщин

В целом, уже можно сказать, что основными факторами модели будет пол и класс пассажира.

Теперь подробнее посмотрим на взаимоотношения между вероятностью выжить и другими факторами. Следующий график подтверждает предположение, что чем выше класс каюты пассажира — тем больше шансы выжить.

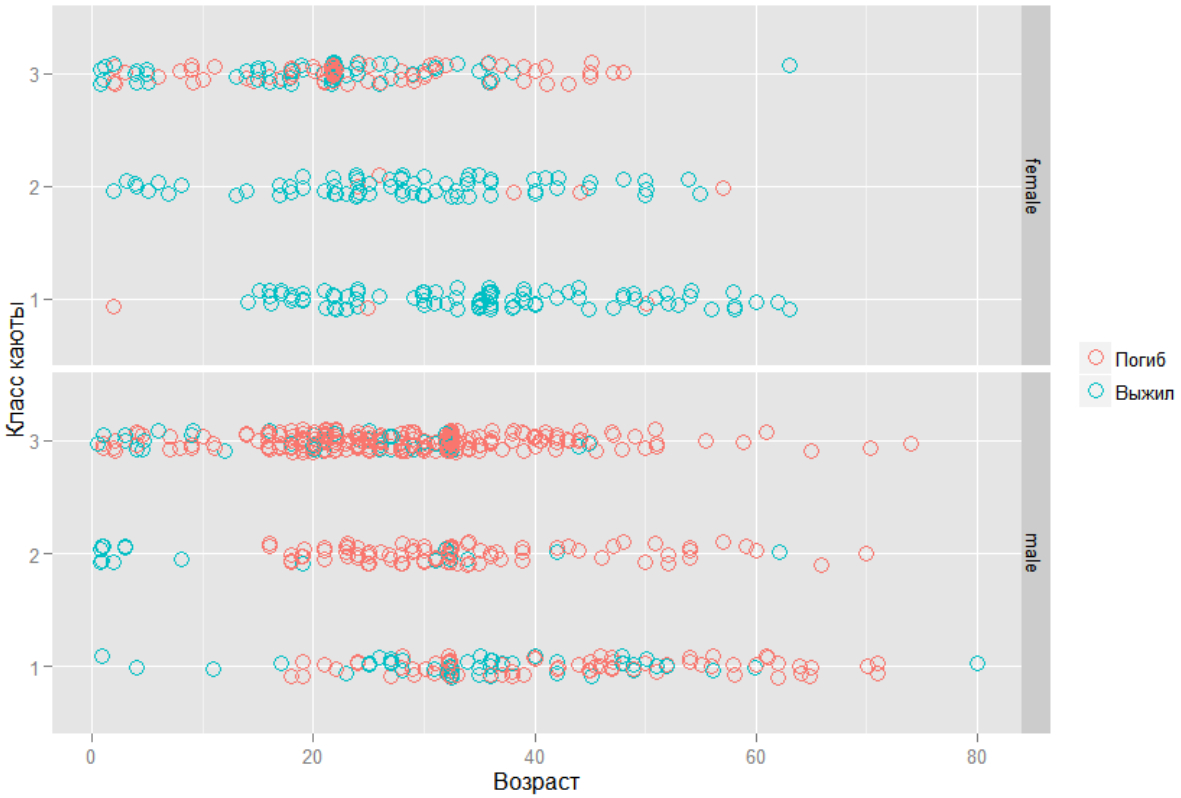


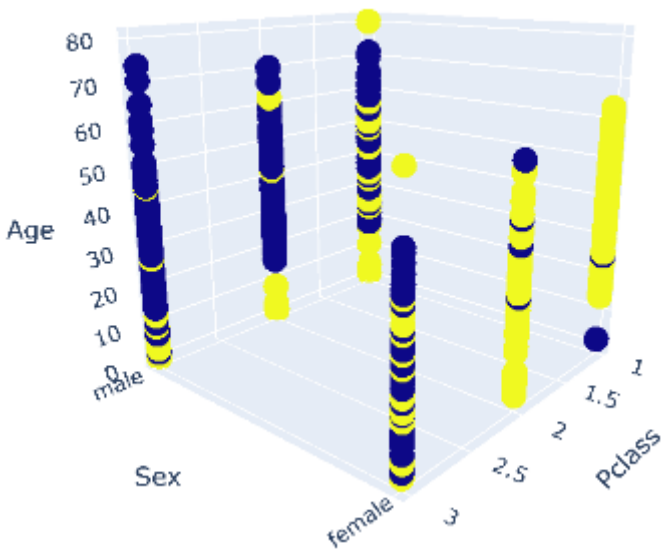
Сравним шансы выжить у мужчин и женщин:



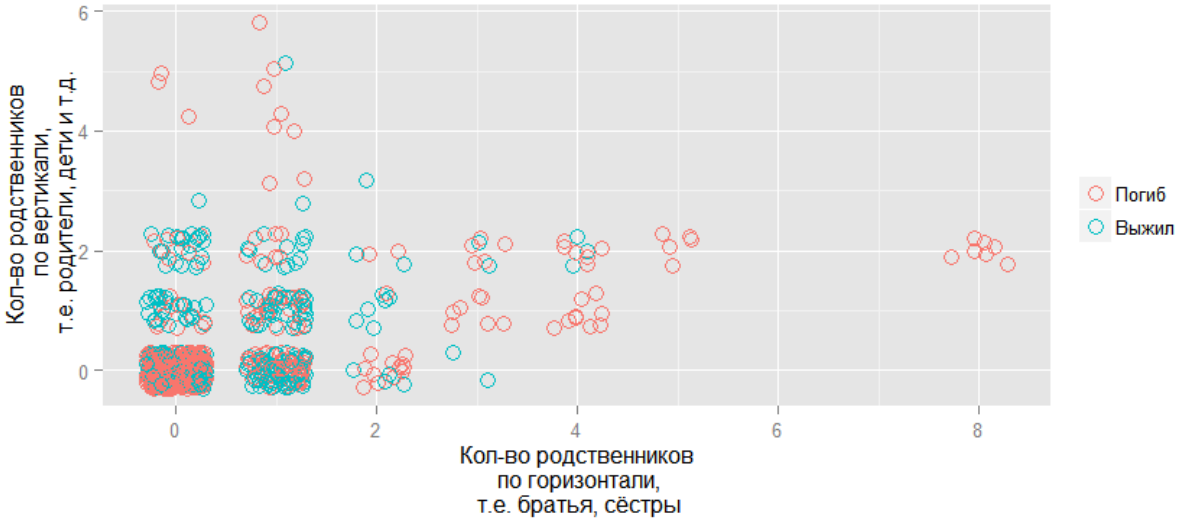
По следующим графикам прекрасно видно, что основные группы выживших — это женщины первого и второго класса всех возрастов. А среди мужчин выжили все мальчики моложе 15 лет кроме третьего класса

обслуживания и небольшая доля мужчин более старшего возраста и в основном из первого класса.





Теперь посмотрим на информацию, которую можно получить из количества родственников на корабле.



Как можно наблюдать, на выживаемость отрицательно влияет как отсутствие родственников, так и большое их количество.

2.3 Подготовка и нормализация

Импортируем необходимые библиотеки

```
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt

# machine learning
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import Perceptron
```

Загружаем данные. В состав предлагаемых данных включены 2 набора:

- Обучающий (train) (содержит 891 примеров):
- Тестовый (test) (содержит 418 примеров).

Их нужно загрузить в локальные переменные. Загружаем обучающий и тестовый наборы данных с помощью библиотеки pandas, она же - pd в нашем примере. Для этого используем стандартные метод .read_csv(..), принимающий на вход путь к файлу с данными.

```
train = pd.read_csv('../input/train.csv')
test = pd.read_csv('../input/test.csv')
```

Конвертируем категориальное значение «*Sex*» в численное представление. 0 - представляет женский пол, 1 - мужской.

```
for dataset in train_test_data:
    dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(int)
```

Создаем на базе «*SibSp*» и «*Parch*» новый признак, отражающий размер семьи - *FamilySize*

```
for dataset in train_test_data:
    dataset['FamilySize'] = dataset['SibSp'] +  dataset['Parch'] + 1

print (train[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=False).mean())
```

```
   FamilySize  Survived
0           1  0.303538
1           2  0.552795
2           3  0.578431
3           4  0.724138
4           5  0.200000
5           6  0.136364
6           7  0.333333
7           8  0.000000
8          11  0.000000
```

Убираем ненужные признаки и сохраняем только те, который будем использовать в эксперименте прогнозирования.

```
features_drop = ['Name', 'SibSp', 'Parch', 'Ticket', 'Cabin', 'Fare', 'Embarked', 'Title']
train = train.drop(features_drop, axis=1)
test = test.drop(features_drop, axis=1)
```

2.4 Обучение и оценка результатов

В этом разделе строим модели предсказания и пробуем разные алгоритмы. Для каждого из применяемых алгоритмов необходимо определить точность его работы на тестовой выборке после обучения на тренировочном наборе. Это даст

нам возможность выбрать наиболее подходящий алгоритм/модель для дальнейшей работы и предсказания финальных результатов.

Данная задача относится к классу задач по классификации данных. Т.к. цель предсказания - определить 0 или 1 в зависимости от того, удастся ли спастись пассажиру или нет. Таким образом имеем задачу бинарной классификации, когда выбора делается из 2 возможных исходов и необходимо отнести запись в тестовом наборе к одному из двух классов - 0 или 1.

Для начала определим тренировочный и тестовые наборы. Поскольку результат вычисления - это (одна) величина, то на выходе алгоритма мы получим вектор ответов y, определяемый по набору передаваемых признаков (это уже матрица) - X. Т.о. y = F(X), такая запись справедлива для любых алгоритмов/моделей. Соответственно вводим переменные (новые, полученный копированием данных из существующих загруженных датасетов train и test - их мы не трогаем!):

X_train - признаки (те, которые мы решили использовать для обучения) из оригинанльного тренировочного набора

y_train - ответы из оригинального тренировочного набора, соответсвующие признакам X_train

X_test - признаки из тестового набора (это новые записи, которые мы НЕ используем в процессе обучения)

y_test - его еще нет, нам как раз и надо его найти

Размерности и структура X_train и X_test должны совпадать, т.к. после проведенного обучения алгоритм F(X) должен оперировать данными одинакового вида.

Существует большое количество алгоритмов классификации. Среди них выбираем следующие для решения поставленной задачи:

- Логистическая регрессия (Logistic Regression);
- Метод опорных векторов (Support Vector Machines (SVM));
- Perceptron (это простейшая нейронная сеть, состоящая из одного нейрона).

Для численной оценки качества алгоритмов используем простой классификатор, который дает доля документов, по которым классификатор принял правильное решение

$$Accuracy = P/N$$

Здесь P – количество документов, по которым классификатор принял правильное решение, N – размер обучающей выборки.

Логистическая регрессия

Logistic regression (or logit regression, or logit model) -- регрессионная модель, где зависимая величиная (DV) является категориальной. Данная статья описывает случай бинарной зависимой величины, где она может применять только 2 значения, "0" и "1", которые представляют ответы вида успешно/ нет, выигрыш/проигрыш, живой/мертый или здоровый/больной. Случаи когда зависимая величина имеет более 2 значений могут анализироваться с помощью мультиномиальной логистической регрессии (multinomial logistic regression), или, если различные категории являются упорядоченными, то с помощью модели упорядоченного выбора (ordinal logistic regression).

Получаем значение Accuracy

```
clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred_log_reg = clf.predict(X_test)
acc_log_reg = round( clf.score(X_train, y_train) * 100, 2)
print (str(acc_log_reg) + ' percent')
```

```
80.47 percent
```

Support Vector Machine (SVM)

Support Vector Machine (SVM), метод опорных векторов - это пример модели обучения с Учителем, используемая для классификационного и регрессионного анализов. Они представляют данные, в виде точек на области, рассатавленных

таким образом, что значения из разных категорий явно разделены между собой промежутком, как можно более широким. Новые значения (сэмплы из выборок) впоследствии соотносятся с этой областью и прогнозируются на принадлежность к какой-либо из категории, в зависимости от того с какой стороны промежутка они находятся.

В дополнение к реализации линейной классификации, SVM может эффективно решать задачи нелинейной классификации, используя специальный прием при работе с ядром, явно соотнося входные значения в много-мерное пространство признаков. Предположим, есть некоторые данные (описываемые в виде точек), принадлежащие одному из двух (возможных) классов, цель - определить к какому из двух классов будет принадлежать (новая) точка или сэмпл. В случае метода опорных векторов, точка отображается как pp-мерный вектор (список из pp индексов), и необходимо узнать можем ли мы разделить такие точки с помощью (p−1)(p−1)-мерной гиперплоскостью.

Алгоритм разделения (кластеризации) который улучшает метод опорных векторов называется - **support vector clustering** и часто используется в промышленном применении.

Получаем значение Accuracy

```
clf = SVC()
clf.fit(X_train, y_train)
y_pred_svc = clf.predict(X_test)
acc_svc = round(clf.score(X_train, y_train) * 100, 2)
print (acc_svc)
```

```
83.39
```

Перцептрон (perceptron)¶

Перцептрон это вид линейного классификатора, т.е. алгоритма классификации, который дает прогноз, основываясь на линейной функции предсказания, комбинирующей множество весов с вектором признаков (свойств).

Значение Accuracy

```
clf = Perceptron(max_iter=5, tol=None)
clf.fit(X_train, y_train)
y_pred_perceptron = clf.predict(X_test)
acc_perceptron = round(clf.score(X_train, y_train) * 100, 2)
print (acc_perceptron)
```

```
76.66
```

Следовательно, для данной задачи максимально эффективным методом оказался метод опорных векторов Support Vector Machine (SVM). Этот метод дает долю правильных решений 0.834. Статистическая ошибка данного результата ~0.03.

3 Вывод

В рамках научно-исследовательской работы за осенний семестр третьего года обучения была проведена работа по изучению нейронных сетей, а также была решена классификационная задача “Титаник”. Для решения данной задачи использовался язык программирования Python.

В процессе данного исследования была изучена теоретическая база как по устройству биологического нейрона, так и по функциям и методам оперирования с математическим нейроном. В качестве примера рассматривалась задача на платформе Kaggle. Изученная теория легла в основу анализа и обработки данных, моделирования графиков зависимостей различных параметров и решения данной задачи с помощью трех различных методов (Логистической регресии, метода опорных векторов, перцетрона).

В результате проведенного моделирования были получены значения точности каждого из методов. С их помощью был определен самый эффективный метод для решения данной задачи “классификации”.