Изучение нейронных сетей и использование их для решения задач классификации

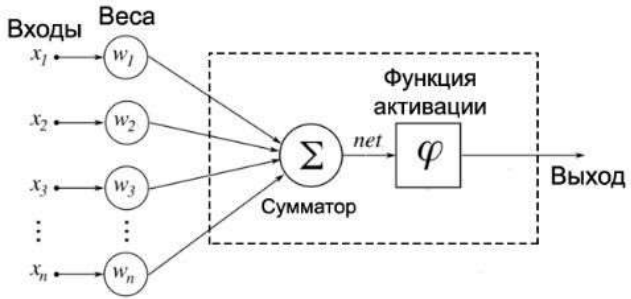
Научный руководитель: д.ф. -м.н. _____А. А. Соколов

Выполнил: _____Т. В. Махкамов

Теоретическое введение

Нейронная сеть

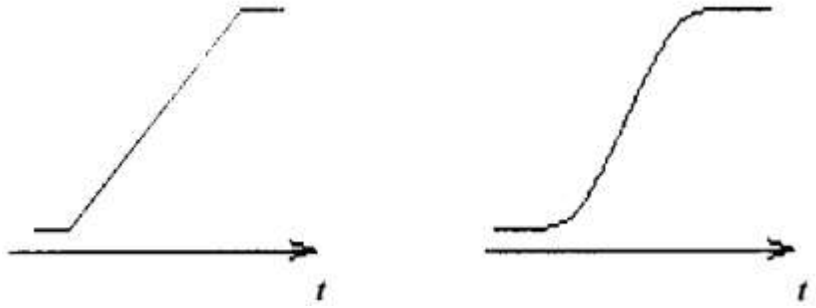
Искусственная нейронная сеть - упрощенная математическая модель биологической нейронной сети. Она представляет из себя последовательность нейронов, соединенных между собой синапсами. Нейрон является базовым компонентом нейронной сети.



Функция активации

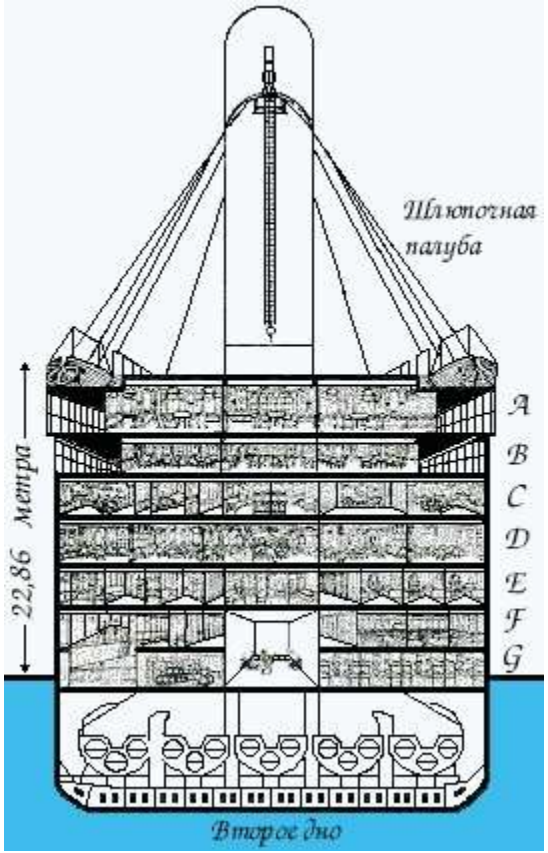
Для формирования выходного сигнала используют функцию активации φ(net), которая преобразует взвешенную сумму в какое-то число, которое и будет являться выходом нейрона.

Для разных типов нейронов используют самые разные функции активации.



Постановка задачи

Задача “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.



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)

Анализ данных для предварительного отбора

Прежде, чем приступить к решению классификационной задачи, необходимо импортировать нужные библиотеки и изучить структуру используемых данных.

```
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
```

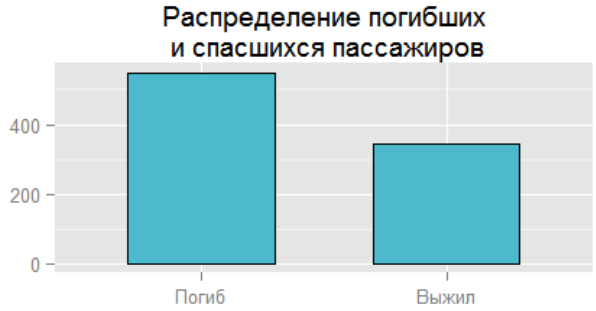
Загружаем данные. В состав предлагаемых данных включены 2 набора:

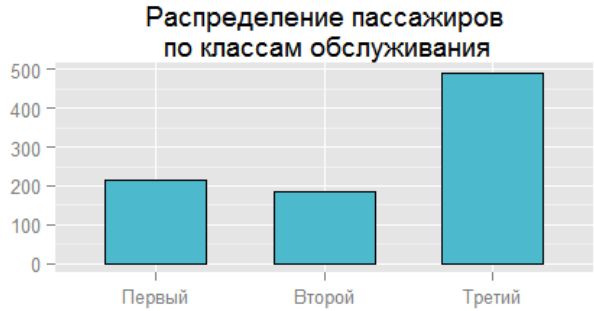
- Обучающий (train) (содержит 891 примеров):
- Тестовый (test) (содержит 418 примеров).

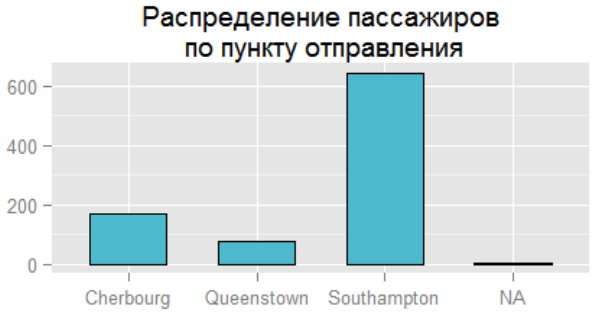
```
train = pd.read_csv('../input/train.csv')
test = pd.read_csv('../input/test.csv')
```

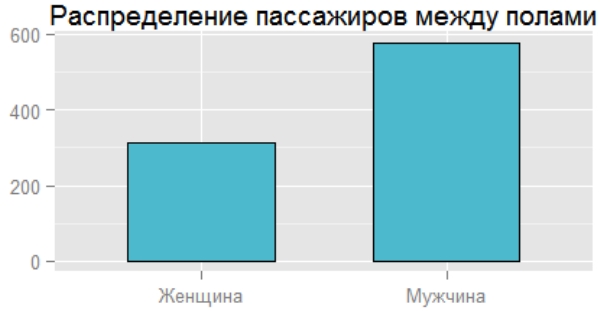
С помощью библиотек seaborn, plotly.express и matploitlib.pyplot, они же - sns, px, plt в нашем примере, соответсвенно, построим различные графики зависимостей.

Анализ данных для предварительного отбора

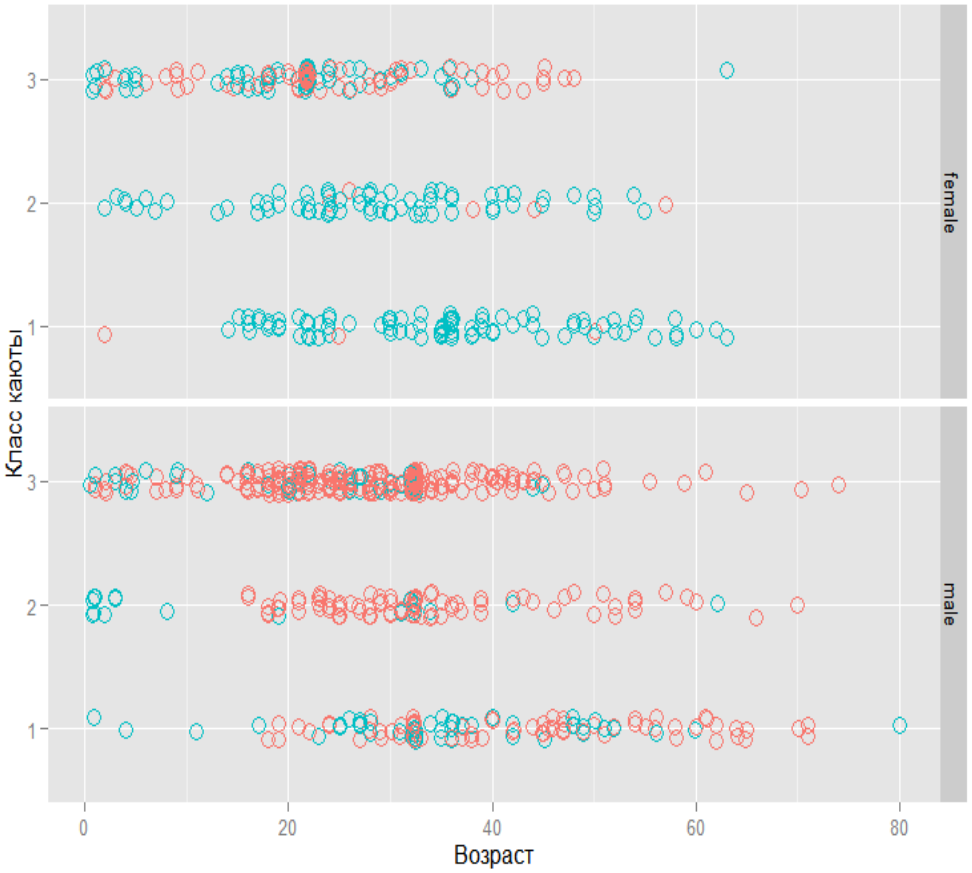


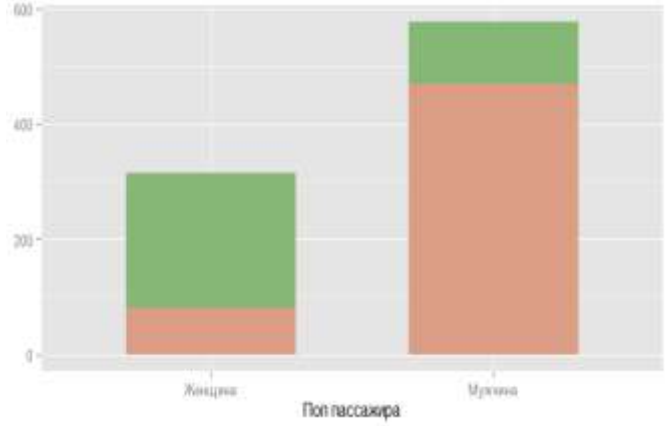


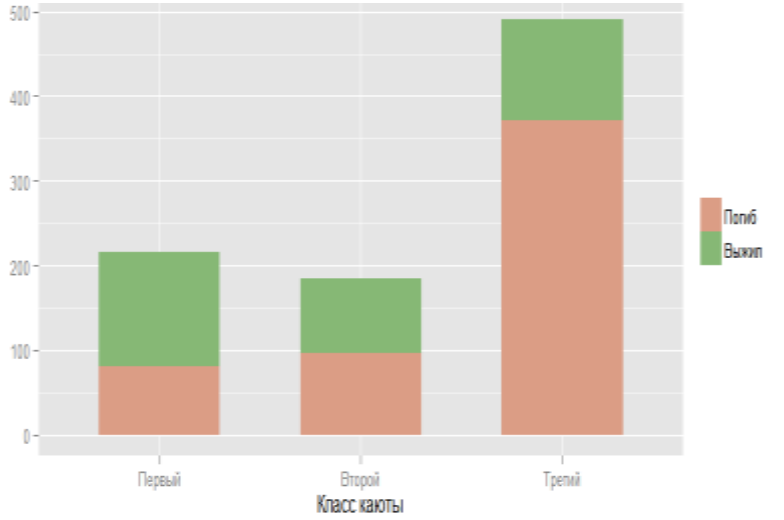




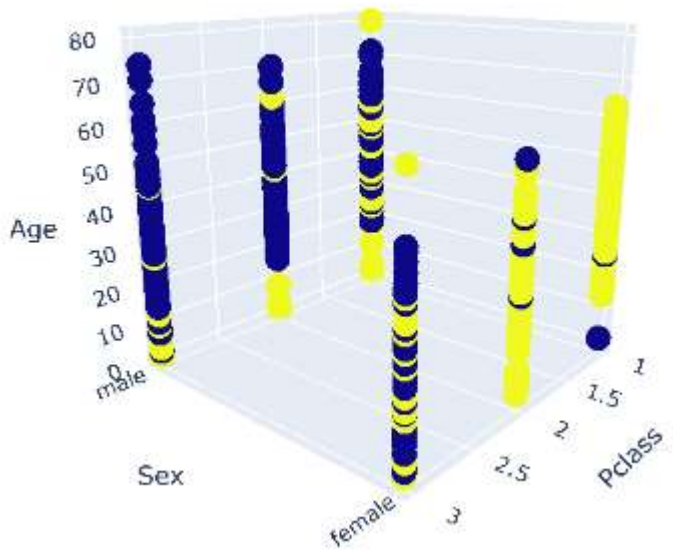
Анализ данных для предварительного отбора

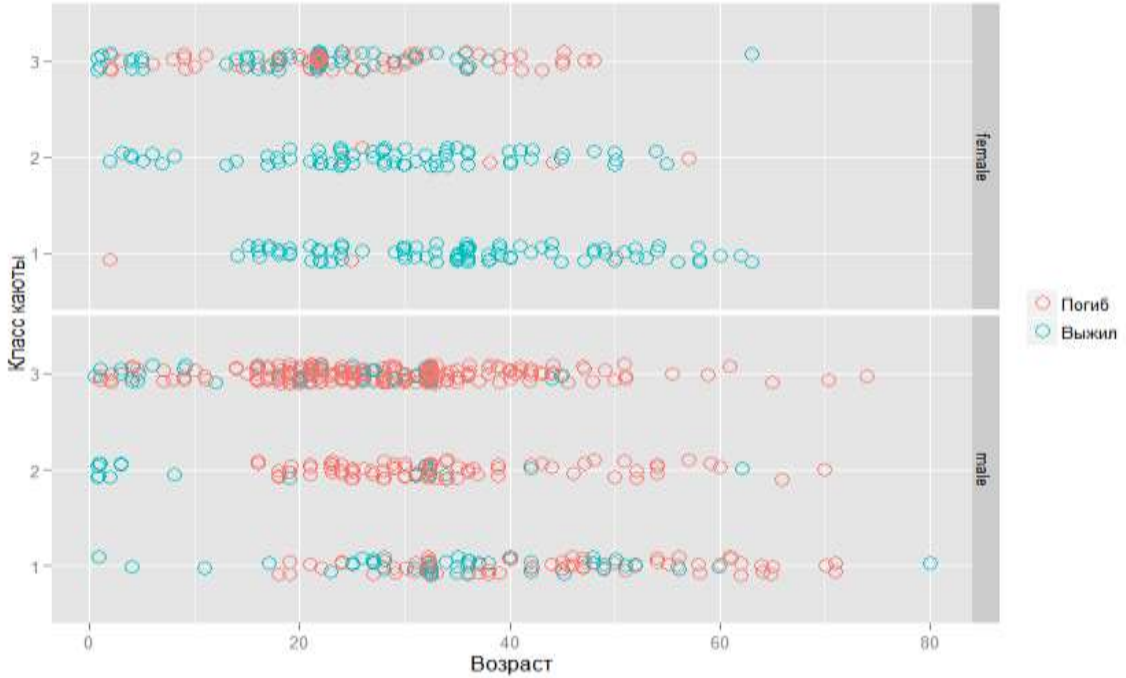






Анализ данных для предварительного отбора





Согласно построенным графикам, можно выделить ключевые категории, это:

“Age” || “Sex” || “Parch” || “SibSp” || “Pclass ||”“Survived”

Подготовим данные

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']

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

	FamilySize	Survived
0	0	0.303538
1	1	0.552795
2	2	0.578431
3	3	0.724138
4	4	0.200000
5	5	0.136364
6	6	0.333333
7	7	0.000000
8	10	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)
```

Классификация и точность

Введем 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.

```
X_train = train.drop('Survived', "PassengerId", axis=1)
y_train = train['Survived']

X_test = test.drop("PassengerId", axis=1).copy()
```

Обучение нейронной сети

Для первой модели обучения выберем - метод Логистической регрессии (Logistic Regression)

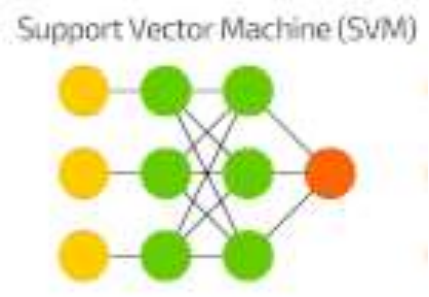
Logistic regression (or logit regression, or logit model) -- регрессионная модель,
где зависимая величиная (DV) является категориальной. Данная статья
описывает случай бинарной зависимой величины, где она может применять
только 2 значения, "0" и "1"

```
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
```

Обучение нейронной сети

Для второй модели обучения выберем - метод опорных векторов (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 наиболее 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.

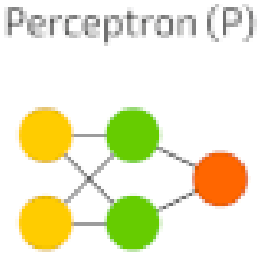
```
clf = SVC()
clf.fit(X_train, y_train)
y_pred_svc = clf.predict(X_test)
acc_svc = round(clf.score(X_train, y_train) * 100, 2)
print (acc_svc)
```

```
83.39
```

Обучение нейронной сети

Для последней модели обучения выберем - персептрон (P)

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




```
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
```

Заключение

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”.

- На основание графиков были выделены, ключевые параметры, используемые для обучения нейронной сети.
- Выполнен анализ данных и построены графики зависимостей различных категориальных зависимостей.
- Данная задача была решена с помощью трех различных методов (Логистической регресии, метода опорных векторов, перцетрона).

Спасибо за внимание!