**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**Московский технический университет связи и информатики**

Контрольное задание

по дисциплине **Интеллектуальные базы данных**

на тему **Прогнозирование результатов футбольных матчей с помощью методов машинного обучения**

для студентов направления 09.04.01 заочной формы обучения

Выполнил студент группы ЗМПП1701

Шутов Леонид

###### Москва 2019

# Оглавление

Оглавление 2

Введение 3

Использованные данные 3

Методы машинного обучения 3

Описание программы 4

Результаты 5

Наивный байесовский классификатор 6

# Введение

В данной работе исследуется применение различных методов машинного обучения к базе данных с информацией о сыгранных футбольных матчей. С помощью методов машинного обучения производятся предсказания о результатах футбольных матчей и сравнивается с реально случившимися результатами. По итогам определяется наиболее походящий и точный для данной задачи метод.

## Использованные данные

Данные, использованные в работе, находятся в открытом доступе

Использовалась база данных с результатами матчей в европейских футбольных лигах с 2008 по 2016 год

Всего около 25000 матчей и статистика по 10000 игроков

## Методы машинного обучения

Использованные методы машинного обучения для предсказания результатов матчей:

Классификатор Случайный лес (количество эстиматоров 200)

Классификатор Адаптивный бустинг (количество эстиматоров 200)

Классификатор Наивный байесовский

Классификатор К соседей

Логистическая регрессия

Классификаторы применялись с использованием Метода главных компонент

# Описание программы

Для реализации был выбран язык Python как один из наиболее простых и часто используемых для задач машинного обучения

Были использованы следующие пакеты:

sqlite3 – пакет для работы с БД SQLite

pandas — программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временны́ми рядами.

Numpy — библиотека с открытым исходным кодом для языка программирования Python. Возможности:

поддержка многомерных массивов (включая матрицы);

поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами.

Sklearn - Scikit-learn - бесплатная библиотека машинного обучения для языка программирования Python.

# Результаты

Полученная точность методов:

Вывод программы:

Score of AdaBoostClassifier for training set: 0.5754.

Score of AdaBoostClassifier for test set: 0.4923.

Score of GaussianNB for training set: 0.5642.

Score of GaussianNB for test set: 0.5538.

Score of KNeighborsClassifier for training set: 0.5698.

Score of KNeighborsClassifier for test set: 0.5077.

Score of LogisticRegression for training set: 0.5419.

Score of LogisticRegression for test set: 0.5077.

Score of RandomForestClassifier for training set: 0.9218.

Score of RandomForestClassifier for test set: 0.5231.

The best classifier is a GaussianNB with a score of 0.5538461538461539.

Таблица с результатами:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод машинного обучения | Параметры запуска | Результат на тренировочном сете | Результат на тестовом сете | Итоговое место |
| Адаптивный бустинг | Количество эстиматоров: 200 | 0.5754 | 0.4923 | 5 |
| Наивный байесовский классификатор |  | 0.5642 | 0.5538 | 1 |
| К соседей |  | 0.5698 | 0.5077 | 4 |
| Логистическая регрессия |  | 0.5419 | 0.5077 | 3 |
| Случайный лес | Количество эстиматоров: 200 | 0.9218 | 0.5231 | 2 |

Наиболее точный результат показал метод Наивный байесовский классификатор:

# Наивный байесовский классификатор

В основе NBC (Naïve Bayes Classifier) лежит, теорема Байеса.

*P*(*c*|*d*)=*P*(*d*|*c*)*P*(*c*)*P*(*d*)P(c|d)=P(d|c)P(c)P(d)

где,

* P(c|d) — вероятность что документ d принадлежит классу c, именно её нам надо рассчитать;
* P(d|c) — вероятность встретить документ d среди всех документов класса c;
* P(c) — безусловная вероятность встретить документ класса c в корпусе документов;
* P(d) — безусловная вероятность документа d в корпусе документов.

Её смысл на обывательском уровне можно выразить следующим образом. Теорема Байеса позволяет переставить местами причину и следствие. Зная с какой вероятностью причина приводит к некоему событию, эта теорема позволяет раcсчитать вероятность того что именно эта причина привела к наблюдаемому событию.

Цель классификации состоит в том чтобы понять к какому классу принадлежит документ, поэтому нам нужна не сама вероятность, а наиболее вероятный класс. Байесовский классификатор использует оценку апостериорного максимума для определения наиболее вероятного класса. Грубо говоря, это класс с максимальной вероятностью.

*cmap*=argmax*c*∈*CP*(*d*|*c*)*P*(*c*)*P*(*d*)cmap=argmaxc∈C⁡P(d|c)P(c)P(d)

То есть нам надо рассчитать вероятность для всех классов и выбрать тот класс, который обладает максимальной вероятностью. Обратите внимание, знаменатель (вероятность документа) является константой и никак не может повлиять на ранжирование классов, поэтому в нашей задаче мы можем его игнорировать.

*cmap*=argmax*c*∈*C*[*P*(*d*|*c*)*P*(*c*)]cmap=argmaxc∈C⁡[P(d|c)P(c)]

Классификатор называется наивным, так как Байесовский классификатор представляет документ как набор слов вероятности которых условно не зависят друг от друга (хотя обычно это не так).

Если на этапе классификации вам встретится слово которого вы не видели на этапе обучения, то значения $W\_{ic}$, а следственно и $P(w\_i|c)$ будут равны нулю. Это приведет к тому что документ с этим словом нельзя будет классифицировать, так как он будет иметь нулевую вероятность по всем классам. Избавиться от этой проблемы путем анализа большего количества документов не получится. Вы никогда не сможете составить обучающую выборку содержащую все возможные слова включая неологизмы, опечатки, синонимы и т.д. Типичным решением проблемы неизвестных слов является аддитивное сглаживание (сглаживание Лапласа). Идея заключается в том что мы притворяемся как будто видели каждое слово на один раз больше, то есть прибавляем единицу к частоте каждого слова.