Предсказание района возникновения ДТП

|  |  |  |
| --- | --- | --- |
| Смирнов Михаил | Чебоксаринов Андрей | Винницкий Владимир |

# Введение

Предсказание района возникновения ДТП по характерным признакам возможной аварии является актуальной задачей для множества социальных организаций, таких, например, как дорожная служба, скорая помощь, а также муниципальное правительство области в общем. Эффективный метод предсказывания района возникновения ДТП позволяет выявить не только наиболее опасные районы для того или иного типа дорожной ситуации, но и определить проблему этой опасности по косвенным причинам, которыми являются входные данные для метода предсказания.

Данная задача решается на выборке, представленной правительством провинции Квинсленд (Австралия) на своем официальном сайте [1]. Для решения задачи применяются Gradient Boosting Trees, Random Forest и Наивный Байесовский Классификатор, реализованные в библиотеке scikit-learn [2] для языка программирования Python.

# Описание данных

Рассматриваемые в рамках данной задачи данные представляют собой историю произошедших ДТП в провинции Квинсленд (Австралия) в периоде 2001 – 2015 гг. Каждый ДТП описан следующими признаками:

* *Crash\_Year* - год, в котором произошло ДТП (2001-2015)
* *Crash\_Police\_Region* – регион провинции (Central, Brisbane, Northern, Southern, South Eastern, Unknown). Значение данного признака необходимо предсказать.
* *Casualty\_Severity* – тяжесть аварии (Fatality, Hospitalized, Minor injury, Medically treated)
* *Casualty\_AgeGroup* – возрастная группа пострадавшего (0 to 16, 17 to 24, 25 to 29, 30 to 39, 40 to 49, 50 to 59, 60 to 74, 75 and over, Unknown)
* *Casualty\_Gender* – пол пострадавшего (Male, Female, Unknown)
* *Casualty\_RoadUserType* – класс пострадавшего с точки зрения дорожного движения (Bicyclist, Passenger, Pedestrian, Driver, Motorcyclist, Other)
* *Casualty\_Count* – количество произошедших ДТП с одинаковыми значениями признаков, описанных выше.

Полная база данных содержит 15744 записи. Из нее формируются тестовые и обучающие выборки.

# Gradient Boosting Trees

## Общие сведения

Boosting – техника машинного обучения для решения задач классификации и восстановления зависимости, которая строит решающую функцию в виде группы простых решателей [3, 4]. Задача обучения с учителем подразумевает наличие обучающей выборки

где – значения входных признаков задачи, – число входных признаков, – выходной признак задачи, который необходимо предсказать по значениям , – объем обучающей выборки.

Для обучения и последующего решения задачи выбирается модель:

где – параметры модели .

Обучение модели заключается в подборе таких значений , при которых построенная решающая функция достигает (локального) минимума функции ошибки на обучающей выборке:

В более общем представлении, задача машинного обучения заключается в нахождении такой решающей функции (X), которая минимизирует математическое ожидание функции ошибки [4]:

Так как зависимость между и в общем случае является стохастической, мы можем только оценить совместную вероятность , из чего следует, что можно лишь аппроксимировать:

Boosting аппроксимирует в виде

где – базовый решатель, обычно представляющий собой простую функцию, – параметры -го базового решателя, полностью содержится в .

Gradient Boosting Trees используют в качестве базового решателя деревья решений небольшой глубины (2-10 уровней), строят решающую функцию итеративно по возрастанию .

Так, изначально строится наилучшая константная модель ( – дерево глубины 1):

Далее итеративно при добавляется новый базовый решатель в общую модель :

По окончании итераций полагается . При этом каждое новое дерево строится таким образом, чтобы добавление данного дерева в общую модель обеспечивало уменьшение ошибки на обучающих данных. Это достигается применением метода градиентного спуска:

* Вычисляется вектор антиградиента ошибки по построенной решающей функции на предыдущем шаге:
* Дерево строится на обучающей выборке . Строится разделение всего пространства векторов из обучающей выборки на непересекающихся областей , в результате чего получается:

где .

* Коэффициент выбирается как

Так как дерево предсказывает константное значение для каждой области разбиения , можно разделить вычисление по этим частям:

В результате чего текущая модель может быть обновлена независимо для каждого региона разбиения :

где – параметр, контролирующий скорость обучения. Эмпирически было показано [4], что малые значения данного параметра () способствуют большей обобщающей способности модели.

Таким образом, можно записать алгоритм обучения Gradient Boosting Trees:

1. Вычисляется константная модель
2. Для всех до :
   1. Вычисляется для всех .
   2. Производится разбиение векторов из обучающей выборки на множеств .
   3. Вычисляется
   4. Новый базовый решатель добавляется в общую модель

.

## Применение GBT к рассматриваемой задаче

Библиотека scikit-learn требует, чтобы данные, используемые для обучения методов машинного обучения, которые в ней реализованы, были представлены в числовом виде. Так как в нашем случае числовыми являются только 2 атрибута (*Crash\_Year, Casualty\_Count*), остальные необходимо преобразовать. Это действие осуществляется двумя способами:

1. Замена каждого значения текущего атрибута уникальным целым числом. Например, для атрибута *Casualty\_Gender* его значения заменяются соответственно на:
   1. Male -> 0
   2. Female -> 1
   3. Unknown -> 2
2. Бинаризация: замена одного атрибута на – количество возможных значений данного атрибута. Каждый из созданных атрибутов является бинарным – т.о. в -м атрибуте содержится информация, является ли значение исходного атрибута равным -му классу. Например, вместо *Casualty\_Gender* будут созданы атрибуты *Casualty\_Gender0 (isMale), Casualty\_Gender1 (isFemale)* и *Casualty\_Gender2 (isUnknown).* Значения исходного атрибута конвертируются следующим образом:
   1. Male -> (1,0,0)
   2. Female -> (0,1,0)
   3. Unknown ->(0,0,1)

В рамках численного эксперимента была предпринята попытка подобрать наилучшее значение количества деревьев в ансамбле при следующих неизменных параметрах GBT [5]:

* *Максимальная глубина дерева* = 5
* *Скорость обучения (learning rate)* = 0.1
* *Процент используемой обучающей выборки на итерации* = 100%
* *Функция ошибки =* логистическая регрессия (deviance)

При этом тестовая и обучающая выборка были сформированы двумя способами:

* «Детерминированная выборка» - тестовая выборка содержит каждую пятую запись исходных данных; обучающая – все остальные.
* «Псевдослучайная выборка» - обучающая выборка есть 20%-ная псевдослучайная подвыборка исходных данных; тестовая выборка содержит все остальные записи.

Таким образом «обыгрываются» сразу 2 сценария обучения:

* обучающая выборка больше тестовой, данные в ней отсортированы по значениям атрибутов лексикографически.
* Обучающая выборка меньше тестовой, данные не отсортированы.

Результаты эксперимента представлены на графиках ниже (см. рис. 1-4). Для вычисления ошибки использовалась Хеммингова функция, реализованная в sciket-learn [6]. Минимум значений ошибки на тестовых данных в обоих экспериментах достигается при максимальном числе деревьев = 50. Значение на детерминированной тестовой выборке (рис. 1) получилось меньше, чем на псевдослучайной (0.7 против 0.72) из-за того, что детерминированная тестовая выборка гораздо меньше по объему, а значит менее репрезентативна. При этом форма записи значений в выборках качественно не влияет на график ошибок (см. рис. 1, 4 и рис. 2, 3 соответственно).

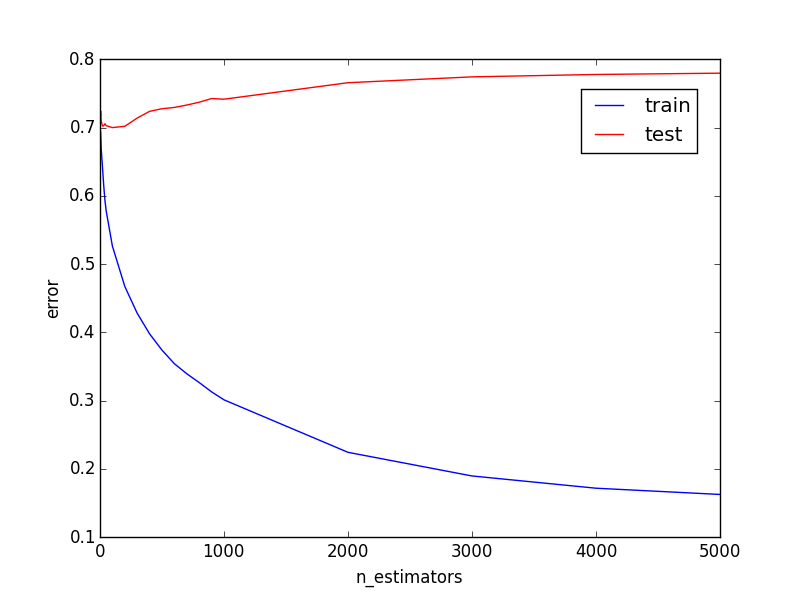


Рисунок 1. Зависимость значения ошибки от роста сложности модели на детерминированной бинаризованной тестовой выборке

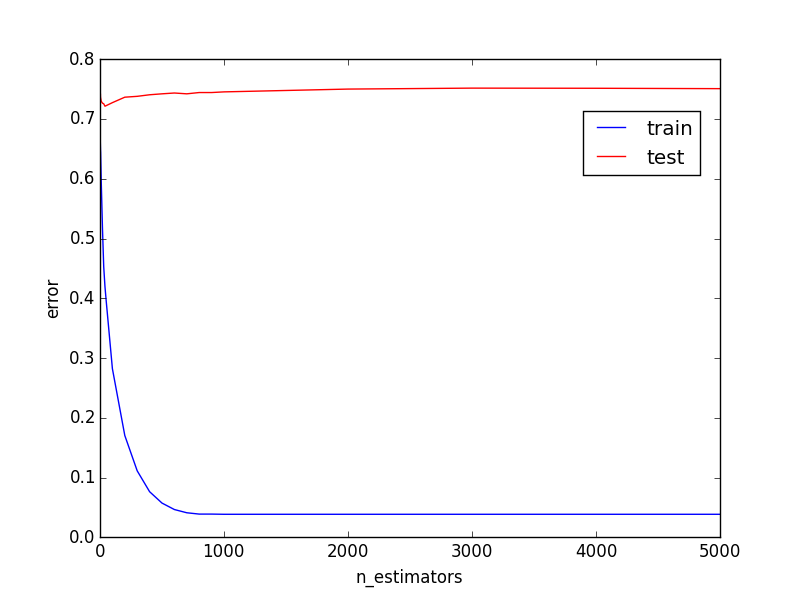


Рисунок 2. Зависимость значения ошибки от роста сложности модели на бинаризованной превдослучайной выборке

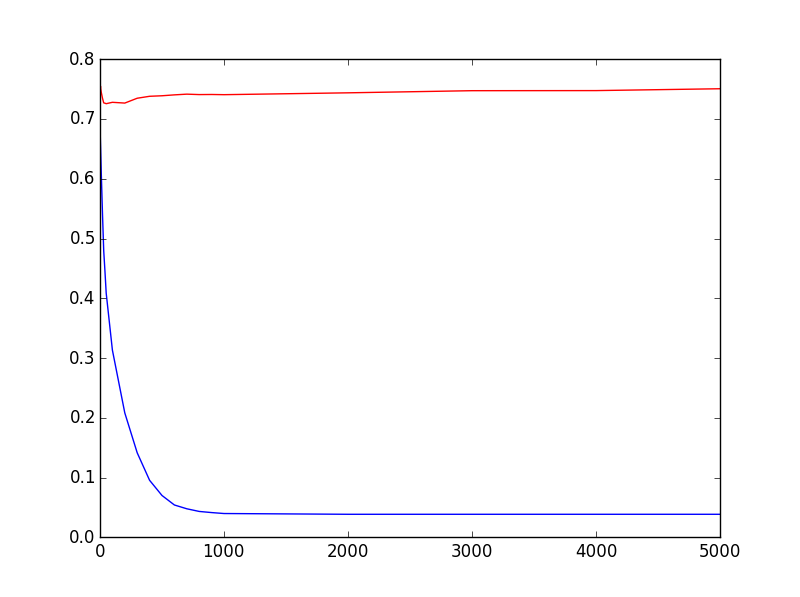


Рисунок 3. Ошибки на небинаризованной псевдослучайной выборке

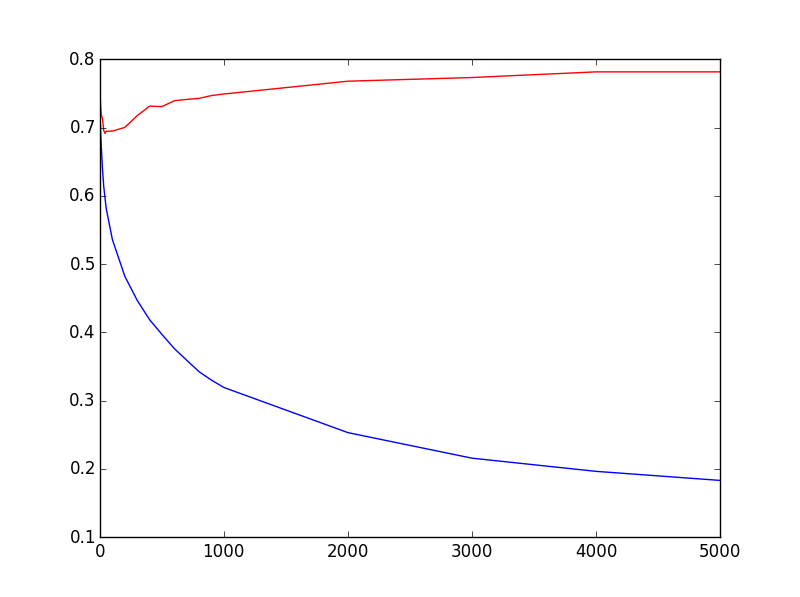


Рисунок 4.Ошибки на небинаризованной детерминированной выборке

Увеличение ошибки на тестовой выборке и уменьшение на обучающей выборке при возрастании сложности модели свидетельствует о том, что модель переобучается. Из этого может следовать, что модель либо не может выявить скрытые закономерности в данных, что можно частично решить переформированием структуры выборок с целью достижения максимальной зависимости между выходным признаком и входными, либо что выходной параметр является случайным – не зависит от значений входных параметров. Второе суждение кажется более правдоподобным, т.к.

1. 4 различных способа построения выборок показали, что результат работы метода не зависит от структуры выборки.
2. Число записей вектора значений входных параметров для каждого значения выходного параметра в общей базе данных примерно одинаково (около 3000 для каждого класса, 300 для неопределенного региона), причем для каждого значения выходного параметра перебираются почти все возможные комбинации входных параметров (число возможных комбинаций всех входных параметров, исключая регрессионный параметр *Casualty\_Count,* равно ).

# Support vector machines (SVM)

## Общие сведения

Support vector machines (машина опорных векторов) – набор алгоритмов машинного обучения, использующихся для обучения с учителем и использующихся как для задач классификации, так и для задач восстановления регрессии. В основе метода лежит идея перевода исходных векторов в пространство более высокой размерности с последующим поиском разделяющей гиперплоскости с максимальным зазором.

В общем случае рассматривается задача обучения по прецедентам , где – пространство объектов, – множество ответов, – целевая зависимость, значения которой известны только на объектах обучающей выборки , . Требуется построить алгоритм , аппроксимирующий целевую зависимость на всём пространстве .

Рассмотрим более узкую задачу – задачу классификации на два непересекающихся класса в которых объекты описаны – мерными вещественными векторами: . Для решения задачи можно построить линейный пороговый классификатор вида:

,

где – скаляр-пороговое значение, - – мерный вектор (параметры алгоритма), а - – мерный вектор, признаковое описание объекта. Уравнение описывает гиперплоскость, разделяющую классы в пространстве . Однако, в общем случае разделяющая плоскость не единственная, поэтому потребуем, чтобы эта гиперплоскость максимально далеко отстояла от ближайших к ней точек обоих классов. Эта плоскость и называется **оптимальной**.

Для нахождения оптимальной гиперплоскости необходимо минимизировать **зазор (margin)** – ширину разделяющей классы полосы, в центре которой и находится разделяющая плоскость. Пусть - две произвольные точки классов, -1 и +1 соответственно, лежащие на границе полосы, тогда ширина полосы описывается как:

и она максимальна при минимальной норме вектора . Другими словами, задача сводится к подбору таких и , при которых минимальна при условии:

которое задаёт полосу между классами.

Данная задача по т. Куна-Таккера эквивалентна двойственной задаче поиска седловой точки функции Лагранжа:

Если и , тогда объект обучающей выборки является **опорным**.

Однако такая постановка задачи возможна только в том случае, если выборка **разделима** (объекты классов в пространстве признаков могут быть чётко отделены друг от друга).

Для того, чтобы решать задачу и на неразделимых выборках, существует 2 способа модифицировать задачу: разрешить классификатору допускать ошибки и ввести функцию штрафа и переход от исходного пространства признаковых описаний к новому, в котором выборка линейно разделима.

Остановимся поподробнее на втором способе. С помощью преобразования переведём исходное пространство в пространство более высокой размерности таким образом, чтобы в выборка была линейно разделима. Такое пространство называется **спрямляющим**. Далее в этом пространстве решаем задачу описанным выше способом, используя вместо старых признаковых описаний новые . Кроме того, в пространстве должно быть определено скалярное произведение (следует из вида задачи поиска седловой точки с заданным штрафом). Формально, скалярное произведение можно заменить **функцией ядра,** котораяявляется функцией и представима в виде при некотором отображении . Как правило, вместо поиска отображения выполняют именно подбор ядра. Примерами ядер являются: полином, радиальная базисная функция (rbf – функция)[7], сигмоидальная функция.

## Применение SVM к рассматриваемой задаче

Для решения задачи используется реализация алгоритма SVM из библиотеки scikit-learn. Описание самой задачи и способы подготовки данных описаны в разделе 2.2. На вход алгоритма подаются следующие мета параметры:

1. C – величина штрафа за ошибку
2. Имя используемого ядра (linear, poly, rbf, sigmoid, precomputed)
3. Степень полинома – ядра (только для полинома)
4. – параметр для rbf, полиномиального и сигмоидного ядра
5. Точность для остановки алгоритма
6. Количество итераций для остановки алгоритма
7. Вес каждого класса
8. Второстепенные параметры для отдельных ядер (свободный член полинома, shrinking heuristic для ускорения процесса оптимизации, размер кэша ядра)

После предварительных экспериментов с выборкой осталось лишь 2 аргумента, существенно влияющих на ошибку на обучающей и тестовых выборках: величина штрафа (1) и параметр ядра (4). Остальные параметры при варьировании влияли на ошибку слабо (<0,01). Поэтому было принято решение проводить эксперименты, варьируя лишь эти 2 параметра и способ подготовки данных. В качестве функции ядра была взята функция rbf, выставленная по умолчанию. Максимальное число итераций – 100000 (при более высоких значениях расчёты выполнялись слишком долго). Остальные параметры (кроме штрафа и ) взяты по умолчанию[8].

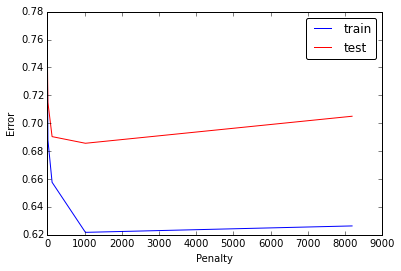
Лучшая комбинация значения штрафа и параметра γ выбиралась с использованием поиска по сетке с экспоненциальным ростом значения штрафа и . Вычислялось значение ошибки для каждой пары значений C и из множеств:

такая проверка позволила вычислить величину ошибки при параметрах разных порядков. Кроме того, каждая пара проверялась на данных с разным способом подготовки и распределением по выборкам (см. раздел 2.2). Для каждой комбинации «способ подготовки/способ распределения» была выделена лучшая пара параметров. Ниже представлена таблица с численными значениями параметров и ошибки для разных способов работы с данными.

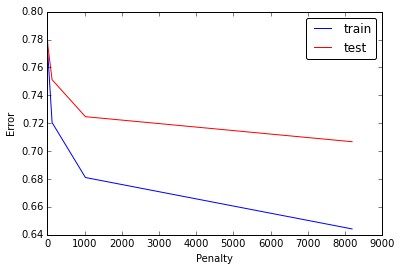
**Таблица 1: Численные характеристики лучших результатов экспериментов**

|  |  |  |  |
| --- | --- | --- | --- |
| Способ работы с данными\параметры классификатора | Значение штрафа | Значение параметра | Ошибка |
| Детерминированная/Бинаризованная |  |  | 0.685 |
| Псевдослучайная/Бинаризованная |  |  | 0.718 |
| Детерминированная/Небинаризованная |  |  | 0.729 |
| Псевдослучайная / Небинаризованная |  |  | 0.741 |
| Детерминированная/ С добавлением ковариационных признаков |  |  | 0.737 |
| Псевдослучайная / С добавлением ковариационных признаков |  |  | 0.740 |

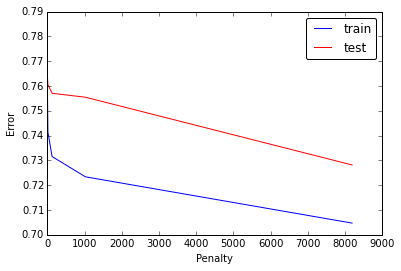
Ниже приведены графики для разных способов работы с данными и лучших значений параметра



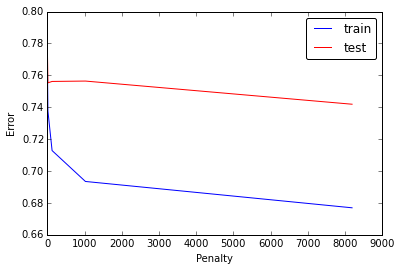
**Рисунок 5: график ошибки для детерминированной, бинаризованной выборки при**



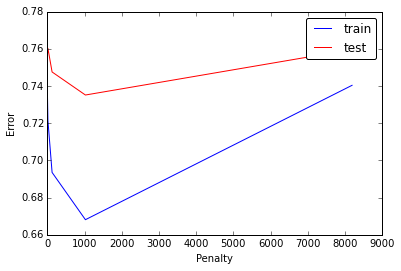
**Рисунок 6: график ошибки для псевдослучайной, бинаризованной выборки при**



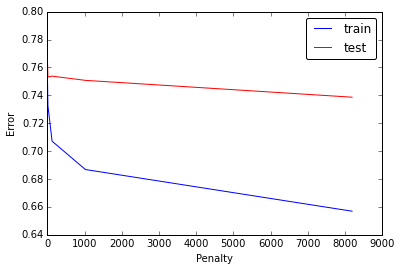
**Рисунок 7: график ошибки для детерминированной, небинаризованной выборки при**



**Рисунок 8: график ошибки для псевдослучайной, небинаризованной выборки при**



**Рисунок 9: график ошибки для детерминированной выборки с добавлением ковариационных столбцов при**



**Рисунок 10: график ошибки для псевдослучайной выборки с добавлением ковариационных столбцов при**

Наименьшее значение ошибки – 0.685 является достаточно высоким. Это, вероятно, обусловлено слабой зависимостью между атрибутами, т.к. варьирование способов подготовки данных, разделения выборки на тестовую и обучающую, а также варьирование параметров метода в широком диапазоне, не дали существенного уменьшения ошибки. Кроме того, другие методы классификации отработали с похожим значением ошибки, что свидетельствует о том, что причина высокого значения ошибки – в самих данных.

# Список литературы

1. [Queensland goverment data. Road Casualties](https://data.qld.gov.au/dataset/crash-data-from-queensland-roads/resource/3fc53539-d529-4c1d-85f8-6c92d9e06fc8)

1. [Scikit-learn](http://scikit-learn.org/)
2. Friedman, J.H. (March 26, 1999). Stohastic Gradient Boosting.
3. Friedman, J.H. (1999). Greedy Function Approximation: a Gradient Boosting Machine. Technical report, Dept. of Statistics, Stanford University
4. [Gradient Boosting Classifier](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier)

1. [Hamming loss](http://scikit-learn.org/stable/modules/model_evaluation.html" \l "hamming-loss)
2. [Radial basis function](https://en.wikipedia.org/wiki/Radial_basis_function)
3. [Skikit SVM API Documentation](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)
4. [Support Vector Machines](http://www.ccas.ru/voron/download/SVM.pdf)