**Казанский (Приволжский) Федеральный Университет**

Отчет по курсу Data Science

Работу выполнил

Хаков Рустам

Высшая школа ИТИС

группа 11-401

## 1. Описание проекта и набора данных

В качестве набора данные я выбрал один из действующих соревнований с ресурса kaggle.com - [Titanic: Machine Learning from Disaster](https://www.kaggle.com/c/titanic).

Задача заключается в том, чтобы предсказать, выживет ли потенциальный пассажир судна в результате крушения корабля или нет, опираясь на тренировочные данные.

Этот набор данных является задачей **классификации с учителем**, так как диапазон значений предсказаний ограничен (имеется два класса вариантов ответа – выживет и не выживет), также есть тренировочный набор данных с известными значениями предсказываемой характеристики (958 значений).

В выбранном наборе данных были следующие данные о пассажирах:

|  |  |  |
| --- | --- | --- |
| Параметр | описание | значения |
| Embarked | порт, в котором пассажир взошел на судно | C Q S |
| Cabin | номер кабины (каюты) | Число |
| Fare | цена билета | Число |
| Ticket | номер билета | Число |
| Parch | число родителей/детей на борту | Число |
| Sibsp | число братьев/сестер/супругов на борту | Число |
| Age | Возраст | Число |
| Sex | пол. Возможные значения: | male, female |
| Name | Имя, фамилия и титул | Строка |
| Pclass | социально-экономический статус кабины пассажира. | 1, 2, 3 |
| passengerId | идентификатор пассажира | Число |
| Survival | Бинарное значение, выжил пассажир или нет | 1, 0 |

Так как большинство методов обучения предполагают работу с числовыми значениями, нужно нормализовать данные

## 2. Преобразования данных

Для качества работы алгоритмов машинного обучения большое значение имеет *выбор атрибутов* и их *нормализация*.

*Выбор атрибутов:*

Первым делом я провел первичный анализ атрибутов. Построил графики зависимости выборочного параметра от результирующего параметра и выявил значения, которые не влияют на качество конечной выборки и соответственно на предсказание. Это *PassangerId.*

Так как остальные параметры могут иметь влияние, стоило выделить те параметры, которые либо слабо влияют (слабо коррелируют с результирующим параметром), либо же влияют, но имеют зашумленные данные. Такими являются *name и cabin*. Эти параметры меньше всего коррелируют с исходным ответом, но параметр name имеет слишком много данных, отделив от этого имени титул пассажира, построив графики зависимости, я выявил, что титул имеет довольно таки большую корреляцию порядка 40-50 %. Поэтому я отделил статус от имени.

Еще одним текстовым параметром остался параметр sex, его меняем на булевы значения 1-male, 0-female

Параметр Age имеет менее 8% корреляции, но она имеет порядка 20% пустых значений, поэтому я решил заполнить их случайными величинами, в зависимости от титула человека, таким образом, я уменьшил влияния пустых значений.

Все остальные имеют корреляцию больше 10%, но некоторые из них я решил немного модифицировать.

Введя новый параметр is\_young, характеризирующий возраст пользователя (если пассажир моложе 12 лет, то 1, если от 12 до 17, то 0.7, если возраст до 45 лет, то 0,2 и если старше, то 0). Так как был введен новый параметр для оценки возраста, то нужно удалить Age

Так же я ввел новый параметр is\_cabin, который характеризирует, имеет ли билет указания кабины.

Вместо параметра SibSp и Parch ввел параметр family, который объединял бы эти два параметра

*Нормализация:*

Я написал несколько функций для нормализации атрибутов, значения которых лежали за пределами диапазона [0, 1]. Это атрибуты *fare, cabin, family*.

Для нормализации титулов я применил функцию, которая перевела бы титулы в числовые значения

Для нормализации Pcalss и Embarked я разделил каждый из них на столбцы булевых функций, так как они имеют только три конечных значения. Получилось 4 колонки Class\_1, Class\_2, C, Q ,остальные столбцы, полученные столбцы имеют коммуникативное свойство, поэтому можно удалить лишние столбцы.

С помощью анализа входных параметров для каждого алгоритма, сумел подобрать параметры, при которых каждый алгоритм работал бы наилучшим образом. Проведя множество экспериментов с типом нейронной сети, пришел к выбору оптимальной.

В процессе обучения так же столкнулся с переобучением данных, когда проходило обучение на нейронных сетях. К счастью данная проблемы была вызвана слишком большим количеством тренировок алгоритма (свыше 300), из-за этого и происходило переобучение.

## 3. Используемые методы машинного обучения

Для решения данной проблемы я использовал следующие методы машинного обучения.

*Метод* ***k*** *ближайших соседей (k-NN):*

Главная идея методазаключается в том, что новый объект присваивается тому классу, который является наиболее распространённым среди соседей данного элемента, где число соседей определяется параметром **k**. То есть этот метод не пытается построить некую обобщенную внутреннюю модель для предсказаний, а просто хранит и использует данные из обучающей выборки.

*Логистическая регрессия (Decision tree):*

Алгоритм построения линейного классификатора, позволяющий оценивать вероятности принадлежности объектов классам. Линейный классификатор — алгоритм классификации, основанный на построении линейной разделяющей поверхности. В случае двух классов разделяющей поверхностью является гиперплоскость, которая делит пространство признаков на два полупространства. Единственной проблемой данной классификации является регуляризация.

*SVC (Метод опорных векторов)*

Метод опорных векторов — набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов. Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.

В качестве ядра используется



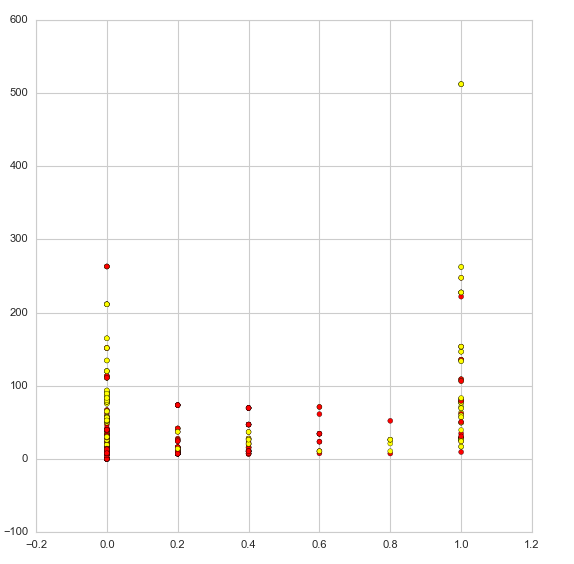
*Случайный лес (Random forest):*

Random forest — алгоритм машинного обучения, заключающийся в использовании сочетания простых решающих деревьев. Алгоритм применяется для задач классификации, регрессии и кластеризации. Оптимальное число деревьев подбирается таким образом, чтобы минимизировать ошибку классификатора на тестовой выборке. В случае её отсутствия, минимизируется оценка ошибки. В отличии от деревья решений случайный лес имеет способность эффективно обрабатывать данные с большим числом признаков и классов. Так же он нечувствителенен к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков. Но в то же время для случайного дерева часто явление переобучения, поэтому нужно тщательно подбирать параметры дерева.

*Искусственные нейронные сети (Artificial neural networks):*

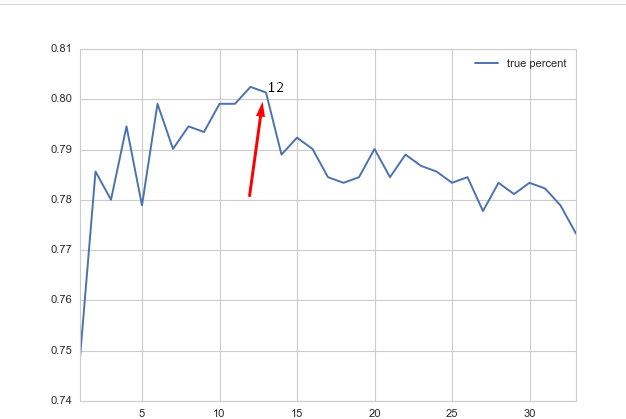
Искусственная нейронная сеть - это математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей (например, мозга человека). Типичная нейронная сеть представляет собой некое множество нейронов, связанных между собой. Принцип работы нейрона не прост, он имеет входной слой данных, один или несколько внутренних слоев, которые преобразуют входные параметры в выходной. Внутренний слой строится на основе какой-либо функции активации, внутренний слой может иметь сложную структуру и состоять из огромного количества внутренних слоев.

4. Для каждого показателя был построен график, показывающий корреляцию признаков, в конечной программе оставил лишь конечную таблицу, состоящую из попарной линейной корреляцией признаков. В конечном листинге кода нету график, которые объясняли бы выбор параметра для каждого из методов машинного обучения.

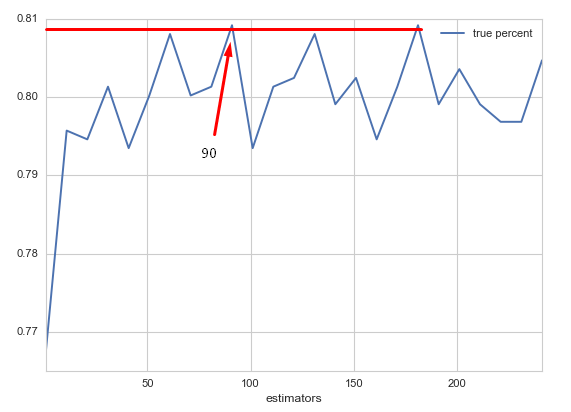
. 

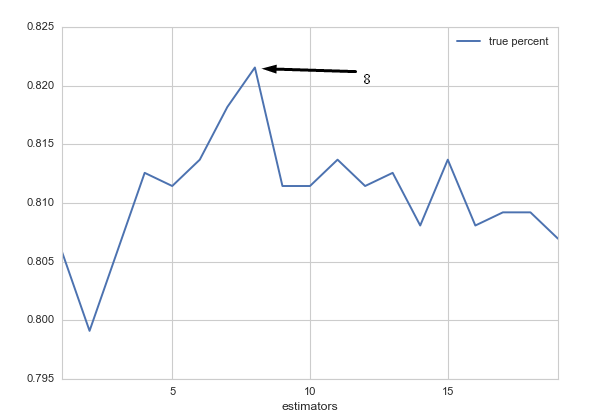
Для работы каждого из алгоритмов нужно было подобрать наилучшие параметры для работы с данными.

Для метода KNN было экспериментальным путем выяснено, что при количестве соседей равным 12, алгоритм работает наилучшим образом.

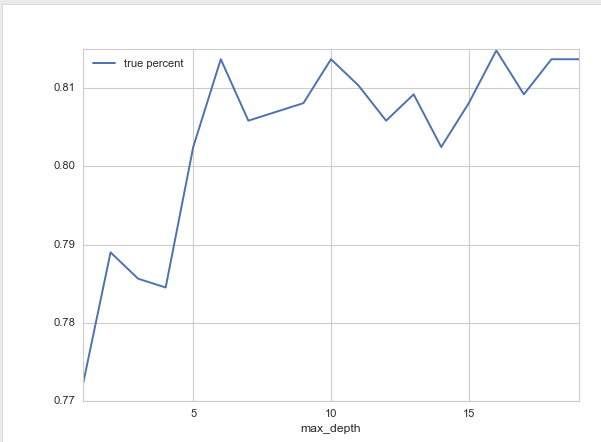


Таким же образом был вычислены наилучшие параметры для RandomForestClassifier, ими оказались n\_estimator=90, min\_samples\_split=8

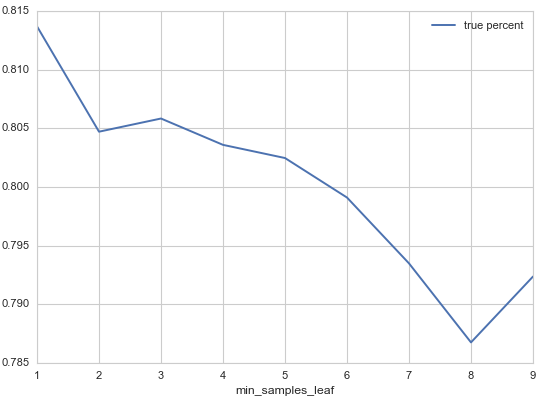




Max\_depth только усугубляет обстановку, поэтому оставляем его как было по умолчанию (None)

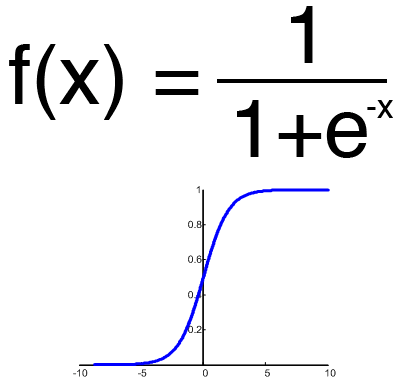


Так же min\_samples\_leaf ухудшается с увеличением параметра, поэтому стоит оставить параметр по умолчанию



Настройка нейронной сети:

Экспериментальным путем было установлено, что сеть работает одинаково хорошо при 4 скрытых секциях и при больших количествах, в качестве функции активации лучше всего показала себя на сигмоиде, вторым по качеству оказался гиперболический тангенс. Это объясняется тем, что гиперболический тангенс используется, как правило для значений, которые принимают как положительные, так и отрицательные значения, в нашем случае все значения положительны.



Следующим шагом была тренировка сети. При итерации модели свыше 300-400 раз был хороший показатель точности на тренировочных данных, но плохой на тестовых. Это означает, что наблюдалось переобучение нейронной сети. Именно поэтому количество итераций

4. Анализ результатов

1. Cross-validation
2. ROC
3. Cross-validation

Данный метод применялся для того, чтобы понять насколько алгоритм работает эффективно на тестовых данных, для данного проекта набор данных был поделен на 3 cross-validation

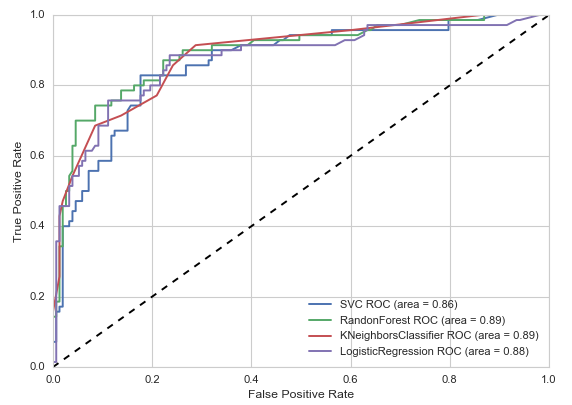
1. ROC-кривая — график, позволяющий оценить качество бинарной классификации, отображает соотношение между долей объектов от общего количества носителей признака, верно классифицированных, как несущих признак. Каждое предсказание может принимать одно из 4 значений:

Если результат классификации *положительный*, и истинное значение тоже *положительное*, то речь идет об *истинно-положительном значении (true-positive, TP)*

Если результат классификации *положительный*, но истинное значение *отрицательное*, то речь идет о *ложно-положительном значении (false-positive, FP)*

Если результат классификации *отрицательный*, и истинное значение тоже *отрицательное*, то речь идет об *истинно-отрицательном значении (true-negative, TN)*

Если результат классификации *отрицательный*, но истинное значение *положительно*, то речь идет о *ложно-отрицательном значении (false-negative, FN)*



## 5. Выводы

В процессе работы над этим проектом я получил представление о методах анализа и решения задач машинного обучения, а именно какие методы нужно использовать для решения задач классификации. Благодаря богатой библиотеке python, понял, какие параметры важны, используя корреляцию признаков и отображение их зависимости на графике.

Результатом машинного обучения был выдан алгоритм с наилучшей точностью на данной задаче классификации. Лучше всего показали себя случайные леса (Random Forest), чуть хуже показали себя нейронные сети, а хуже всего - метод k-ближних соседей. Это обусловлено тем, что случайный лес был настроен качественно, к тому же он не чувствителен к количеству параметров, и он отлично подходит для задач бинарной классификации. Нейронная сеть показала себя хуже, потому что для ее корректной работы довольно трудно подобрать нужное количество тренировок, которое даст хорошую матрицу перехода, а при больших количествах прогона возникает переобучение. Метод k-ближних соседей показывает результат лучше остальных только когда много параметров, сильно коррелирующих с ответом, но таких параметров было очень мало в исходной выборке. Метод опорных векторов показал средние результаты и не претендовал за звание лучшего метода для решения задачи, так как часть данных трудно разделить гиперплоскостью, к тому же алгоритм не работает без качественной регуляризации.