**Большая домашняя задания по УМвАД Атахужаев Собитхон**

Программа состоит из следующих частей:

* Ищем оптимальный порог для улучшения точности нашей классификации и смотрим зависимость точности от нашего порога
* Рассматриваем зависимости насколько точность классификации зависит от объема наших тренировочных данных
* Берем исходные данные с сайта и сами создаем обучающую выборку и тестовую выборку чтобы распределить их равномерно(чтобы их количество были схожи)

**BHW\_Atakhujaev\_Sobitkhon.py**

1. **Алгоритм1**

* Наша программа в цикле изменяет значение порога и для каждого из значений порога запускает внутри себя несколько раз программу «test\_implication\_changed.py», которая внутри себя считает количество правильно и неправильно отгаданных(правильные положительные, правильные отрицательные, неправильные положительные, неправильные отрицательные) классификаций и храним их в списке. После этого из всех полученных точностей выбираем самый максимальный и для полученных результатов считаем Метрики качества и запоминаем лучший порог.

**Алгоритм2**

* Далее для лучшего найденного порога опять запускаем раз программу «test\_implication\_changed.py», которая внутри себя считает количество правильно и неправильно отгаданных(правильные положительные, правильные отрицательные, неправильные положительные, неправильные отрицательные) классификаций и храним их в списке но в этот раз мы в делаем это для 1,2….файлов. Далее находим максимальную (TP+TN) и запоминаем количество файлов при которых мы получили данный результат.

При нашем максимальном пороге получили Accuracy : 0.9279749478079332 когда

* **Алгоритм3**

Чтобы улучшить результат взяли исходные с помощью StratifiedShuffleSplit , пробовали разделить обучающую выборку и тестовую выборку(33%) и для них применили алгоритм машинного обучения **Random forest** (RandomForestClassifier), подсчитали метрики. Прогоняли еще раз метод RandomForestClassifier с наилучшими параметрами, подобранные опытным путем. Потом проверили полученную модель с помощью скользящего контроля и для этого мы воcпользовались функцией [cross\_val\_score](http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.cross_val_score.html" \l "sklearn.cross_validation.cross_val_score):

Accuracy у нас получился 0,991.

**МЕТРИКИ**

Чтобы более понятно было каждая метрика рассмотрим простой пример и опишем каждую из метрик(Нам надо поймать терроситов).

|  |  |  |
| --- | --- | --- |
|  | TRUE | FALSE |
| TRUE | TP | FN |
| False | FP | TN |

TP-поймали террористов правильно

FN – количество пропущенных террористов (пропустили террориста)

FP – количество ложных тревог

TN- количество правильно не потревоженных мирных жителей

* True Positive Rate -  - “всех ли террористов поймали? Или процент правильно определенных террористов”
* True Negative Rate - “процент правильно определенных мирных жителей среди мирных”
* Negative Predictive Value “процент правильно пропущенных мирных жителей среди пропущенных”
* False Positive Rate-“процент потревожанных мирных ”
* False Discovery Rate-“процент мирных среди предсказанных террористов”
* Accuracy-“процент правильных ответов”
* Precision «сколько процентов террористов среди подозреваемых»
* Recall «»
* F1  «Средневзвешенное между precision и recall»

**Алгоритм 1**

Самый максимальный результат был достигнут при пороге равной 1.0 и accuracy было равно 0.9183884297520661.

**Алгоритм 2**

Самый максимальный результат был достигнут при пороге равной 1.0 и accuracy было равно 0.9279749478079332

**Алгоритм 3**

True Positive : 99

True Negative : 187

False Positive : 1

False Negative : 1

True Positive Rate(TPR) : 0.99

True Negative Rate(TNR) : 0.994680851064

Negative Predictive Value(NPV) : 0.994680851064

False Positive Rate(FPR : 0.00531914893617

FDR : 0.01

Accuracy : 0.993055555556

PPV(precision) : 0.99

F1 : 0.99

Recall : 0.99