**Отчет с результатами сравнения моделей для соревнования на Kaggle  
Data science Course Homework /Сравнение классификаторов/**<https://www.kaggle.com/c/jds3>

Таким образом, мы осуществили следующий порядок действий:

* прочитали данные;
* применили KMeand для кластеризации данных и добавили в данные новый признак - получившийся класс;
* с помощью KernelPCA с косинусным ядром снизили размерность;
* обучили набор моделей;
* на основании лучших моделей построили голосующий классификатор;
* к лучшему типу модели применили поиск параметров по сетке;
* прочитали тестовые данные;
* на основании полученных параметров сделали кластеризацию искомых данных и сохранили их в файл.

Самыми лучшими оказались алгоритм SVM и KNeighborsClassifier. Это объясняется тем, что отображение значений пикселей в номер буквы связано сложной нелинейной функцией, но не связано напрямую линейно. Как следствие номер сложно аппроксимировать через простую линейную функцию, и именно поэтому наибольшие результаты дают нелинейные алгоритмы, а SVM с ядром “rbf” показывает результаты намного лучше, чем LinearSVC.

При этом эксперименты, не включенные в отчет, показали, что применение масштабирования данных чаще всего либо немного ухудшало либо немного улучшило результат, буквально на несколько процентов туда-сюда, но сами вычисления производились в разы дольше, в результате чего оно было опущено.

KNeighborsClassifier показал результат совсем не намного хуже SVM, что можно объяснить тем, что, несмотря на то, что мы предварительно проводили преобразование по выявлению и составлению основных фич метод главных векторов с косинусным преобразованием, что существенно снизило количество фич и преобразовало данные под некоторые алгоритмы.

Классификатор дерева решений DecisionTreeClassifier строит всего одно дерево, и, как следствие, сильно переобучается под текущую обучающую выборку. В результате его показатели оказались не очень высокими.

RandomForestClassifier существенно снизил ошибку применения DecisionTreeClassifier, поскольку использует не одно дерево, а целый лес, тем самым минимизируя возможность переобучения одного дерева, однако пакет из коробки без дополнительной настройки не дал идеального результата (при этом в тестовых экспериментах с использованием различных параметров он показывал себя лучше и использовался как один из итоговых вариантов для тестирования моделей).

Далее мы уточняли параметры и результаты подобранных алгоритмов, в начале на крупной сетке, а затем с помощью HalvingRandomSearch и c помощью голосующего ансамбля VotingClassifier, которому в качестве участников отдали простые алгоритмы показавшие наилучший результат с весами, пропорциональными показанному результату.

В результате получили следующие лучшие алгоритмы

* VotingClassifier soft - 0.927252
* VotingClassifier hard - 0.925450
* HalvingRandomSearchCV SVC - 0.925000
* SVC rbf C=5 - 0.913329
* SVC rbf C=6 - 0.913245

Лучшим оказался голосующий ансамбль VotingClassifier с мягким голосованием, поскольку он может учитывать не просто результаты входящих в него моделей, но и вероятности получения данных результатов. Соответственно, жесткое голосование показало результаты немного хуже, однако времени на вычисления ушло намного больше. Поиск по сетке с отбором показал результат совсем немного хуже данных голосующих алгоритмов, поскольку мы отбирали модель с максимальным результатом наиболее эффективного алгоритма, в то время как голосующие алгоритмы учитывали голоса разных моделей, не только самых эффективных.

Лучшую на тестовых экспериментах модель обучили на полном наборе данных и через предсказание на искомых данных получили итоговый результат.