Описание попытки создать классификатор мужских и женских голосов

Анализ данных.

В работе использованы данные LibriTTS\train-clean-100. Они содержат 33 236 фрагментов речи (15 607 мужских и 17 629 женских) от 247 чтецов, из них 124 мужских голосов и 123 женских.

Для того, чтобы получить достаточное разнообразие голосов, был выбран довольно большой датасет, но чтобы сократить время обработки данных, было использовано ограничение по количеству файлов на каждого говорящего (в итоговой версии три файла, это параметр MAX_EXAMPLES_PER_READER = 3 в файлах extract_features_...), итого 741 wav файл.

Выбор метода.

На лекциях говорилось, что основная характеристика, разделяющая мужские и женские голоса – это фундаментальная частота. Но также важную информацию несут и с мел-частотные кепстральные коэффициенты. Я решила посмотреть и на то, и на то (а потом и по отдельности).

Сравнивается пять моделей: Logistic Regression, KNN, Random Forest, Gradient Boosting Classifier и SVM.

Описание эксперимента.

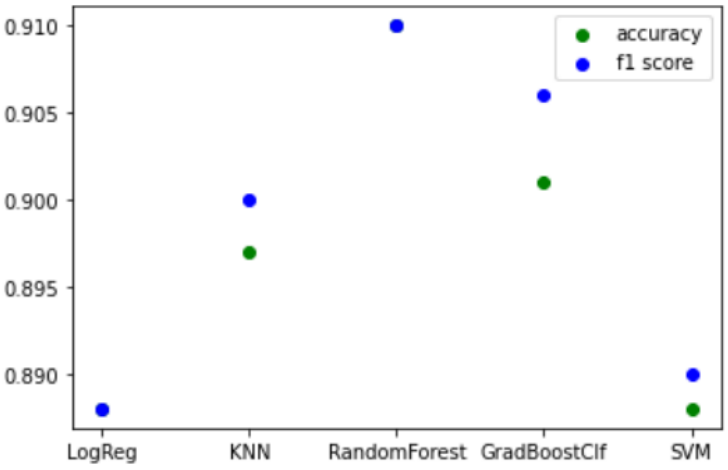
В работе использовались шесть признаков, связанных с фундаментальной частотой (['f0_mean', 'f0_std', 'f0_25p', 'f0_median', 'f0_75p', 'f0_iqr']) и 52 признака с мел-частотными кепстральными коэффициентами (для 13 коэффициентов считается их среднее и дисперсия, а также считается дисперсия их дельт первого и второго порядка, всего 4 * 13 = 52 признака). Итого 58 признаков.

Поскольку извлечение фичей – процесс, требующий существенного времени, я запустила его заранее и сохранила результат в файлах df_with_f0.csv и df_with_mfcc.csv. Файл train_model.py не считает фичи заново, а подгружает из этих файлов.

При необходимости поменять данные и пересчитать фичи, нужно запустить файлы extract_f0_features.py и extract_mfcc_features.py, поменяв в них путь к данным и ограничение по числу файлов на чтеца, если нужно.

В train_model.py содержится обучение и валидация моделей. Поскольку важно, чтобы валидация проводилась на тех голосах, которые модель еще не слышала, train-test split проводится не по аудиофайлам, а по чтецам (причем отдельно по мужчинам и женщинам, чтобы обеспечить сбалансированность).

Вот результаты на validation set при validation ratio = 0.25 (результаты меняются от запуска к запуску, если не фиксировать random seed при разбиении чтецов на train / validation set).



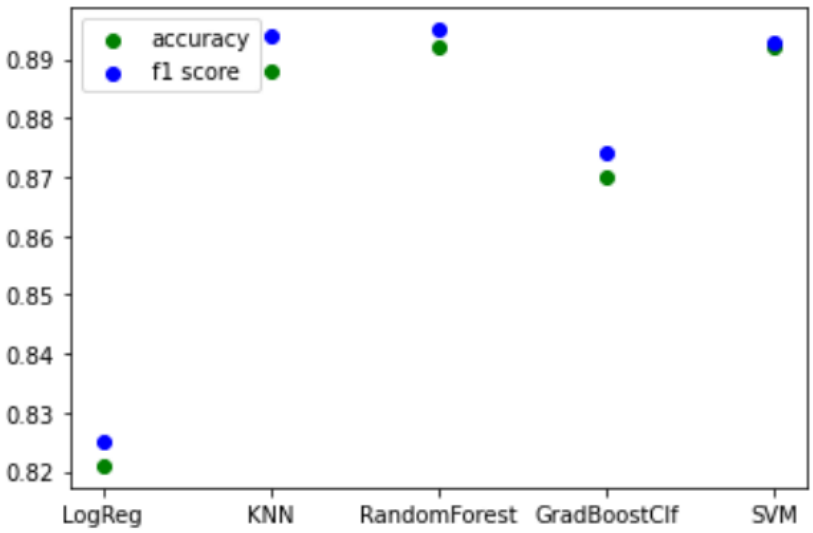

```
LogReg :
Accuracy score =  0.888
F1 score =  0.888

KNN :
Accuracy score =  0.897
F1 score =  0.9

RandomForest :
Accuracy score =  0.91
F1 score =  0.91

GradBoostClf :
Accuracy score =  0.901
F1 score =  0.906

SVM :
Accuracy score =  0.888
F1 score =  0.89
```

Видно, что результаты в целом похожие, но предпочтительнее скорее Random Forest.

Итоговый скор модели – что-то около 90-91%.

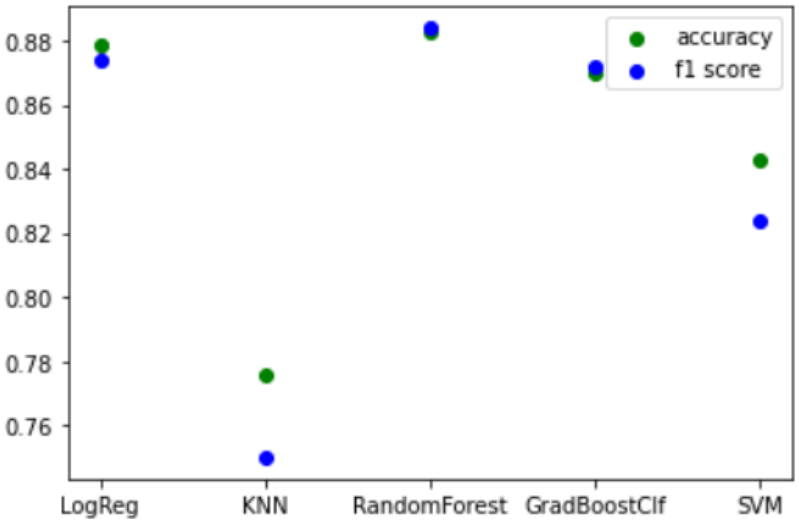
Отдельно интересно посмотреть, как распределяется предсказательная сила между двумя группами данных: f0-related и MFCC-related. Для этого модели прогоняются также на двух отдельных датасетах – только с f0 и только с MFCC.

Результаты на F0-related features




```
LogReg :
Accuracy score =  0.821
F1 score =  0.825

KNN :
Accuracy score =  0.888
F1 score =  0.894

RandomForest :
Accuracy score =  0.892
F1 score =  0.895

GradBoostClf :
Accuracy score =  0.87
F1 score =  0.874

SVM :
Accuracy score =  0.892
F1 score =  0.893
```

Результаты на MFCC-related features




```
LogReg :
Accuracy score =  0.879
F1 score =  0.874

KNN :
Accuracy score =  0.776
F1 score =  0.75

RandomForest :
Accuracy score =  0.883
F1 score =  0.884

GradBoostClf :
Accuracy score =  0.87
F1 score =  0.872

SVM :
Accuracy score =  0.843
F1 score =  0.824
```

Видно, что основная часть объясняется фундаментальной частотой, но тем не менее MFCC немного улучшают точность.

Дальнейшие возможные шаги.

Кажется, стоило бы отправить MFCCs в нейросеть, причем попробовать кроме полносвязной еще варианты, описанные на последней лекции по биометрии. Но для того, чтобы в этом разобраться мне пока не хватило знаний и времени.