Жамков Никита

Метамодели

Начало

Я провел небольшое исследование на простом датасете (Customer Telecom Churn from **Kaggle**), для которого решается задача *бинарной классификации* (1 или 0). Ради задания я не проводил тяжелый анализ или обработку данных, так как, логично, что на необработанных данных, что на обработанных данных – точность будет изменяться прямо пропорционально. К примеру:

Если на необработанных данных у первой модели точность – 0.75, у второй 0.60, а на обработанных данных у первой – 0.89, у второй – 0.74, то пропорция увеличения точностей осталась прежней.

(*если изменения и будут наблюдаться – то незначительные*)

Исследование

Я решил разделить датасет на 20% к 80% и применить RobustScaler к данным для обучения (X). Для проверки точности каждой выбранной модели из списка:

- LogisticRegression
- RandomForestClassifier
- ExtraTreesClassifier
- KNN

Вставил каждую модель в свой модуль для определения точностей (GMS) каждой из модели на валидационной выборке, результаты оказались следующими:

```
Models (test)              Metrics (test)
-------------------------  ------------------------------------------------------------------
LogisticRegression()       {'accuracy': 0.8740629685157422, 'f1-score': 0.3}
RandomForestClassifier()   {'accuracy': 0.9385307346326837, 'f1-score': 0.7544910179640718}
ExtraTreesClassifier()     {'accuracy': 0.9310344827586207, 'f1-score': 0.7124999999999999}
KNeighborsClassifier()     {'accuracy': 0.904047976011994, 'f1-score': 0.5294117647058824}
```

Далее, я решил создать pipeline (*sklearn.pipeline.Pipeline*) для каждой модели, чтобы автоматически применять, сначала, RobustScaler, а потом тренировать модель и/или получать predict.

[101]:
```
for model in models:
    pipe = Pipeline([
        ('scaler', RobustScaler()),
        (str(model), model)
    ])

    pipe.fit(X_tr, y_tr)

    X_tr[str(model)] = pipe.predict(X_tr)
    X_t[str(model)] = pipe.predict(X_t)
```

После применения pipeline-ов, наш датасет выглядит следующим образом:

```
X_tr
```

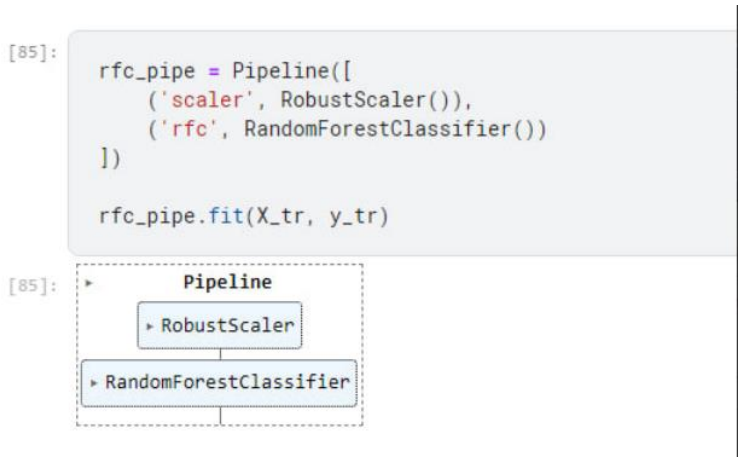
[84]:

ntWeeks	ContractRenewal	DataPlan	DataUsage	CustServCalls	DayMins	DayCalls	MonthlyCharge	OverageFee	RoamMins	LogisticRegression()	RandomForestClassifier()	ExtraTreesClassifier()	KNeighborsClassifier()
80	1	0	0.00	3	198.1	160	47.0	7.84	9.3	0	0	0	0
28	1	0	0.00	3	168.2	87	43.0	8.09	10.1	0	0	0	0
120	1	0	0.00	2	252.0	120	56.0	7.51	9.6	0	0	0	0
105	1	0	0.00	1	251.6	88	58.0	8.76	5.4	0	0	0	0
134	1	1	1.65	2	247.2	105	78.5	11.28	6.1	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
27	1	0	0.00	1	72.7	75	31.0	10.43	9.9	0	0	0	0
89	1	1	1.59	0	97.8	98	50.9	10.36	5.9	0	0	0	0
93	0	0	0.00	1	131.4	78	42.0	10.99	11.1	0	1	1	1
91	1	0	0.00	3	189.3	100	53.0	11.97	9.9	0	0	0	0
130	0	0	0.00	5	216.2	106	68.0	18.19	16.9	1	1	1	1

14 columns

Можно заметить, что появились колонки справа, которые принимают значения либо 0, либо 1. Это результат predict-ов моделей из списка, указанного раннее.

Далее, я решил создать очередной pipeline, только моделью будет являться лучшая модель, проявившая себя на нашем датасете – это RandomForestClassifier, с точностью *0.75449101 F1-score.*



Результаты оценки точности получились следующими:

- Accuracy: 0.938530734
- F1-score: 0.751515151

Еще раз обратим внимание на таблицу из оценок точности КАЖДОЙ модели, проверенной в модуле GMS. Можно понять, что точность F1-score незначительно ухудшилась.

Появляется гипотеза: *«…если мы используем RandomForestClassifier, как главную модель для оценки, конечно же эта модель будет иметь такую же точность, как RandomForestClassifier отдельно!*»

Проверим гипотезу на следующей странице…

Я решил, в качестве модели для независимой оценки, использовать модель, которой не было в **списке.** Этой моделью стал:

Light Gradient Boosting Machine Classifier (*или* LGBMClassifier)

Проделав те же самые операции и построив новый pipeline с новой моделью независимой оценки, получаем следующий результат:

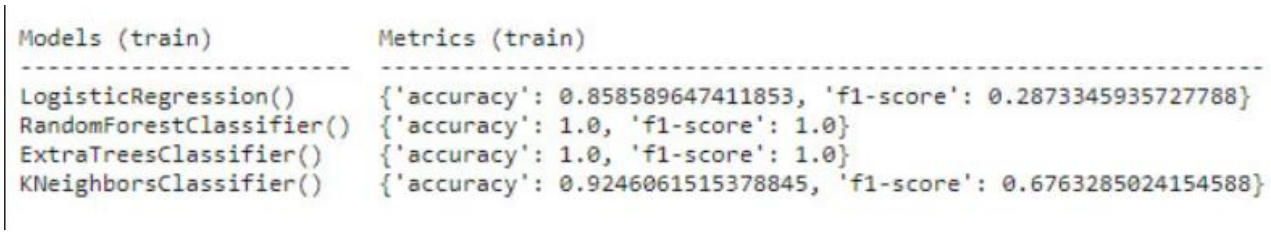
- Accuracy: 0.938530734
- F1-score: 0.751515151

Точность такая же абсолютно, как и в прошлый раз.

Прошлый pipeline:

Новый pipeline:

После некоторого времени рассуждений, можно понять, почему так получается. GMS модуль проанализировал не только валидационную выборку, но и тренировочную, и собрал точности каждой модели во время тренировки.



Можно заметить, что лучшая модель RandomForestClassifier переобучилась на тренировке. Так как *Accuracy и F1-score* точны на 100%, так как это практически невозможно.

Теперь можно рассмотреть ситуацию со стороны LGBM Classifier. В теории, у каждой колонки есть свой параметр “Importance” (с англ. *важность*). Модель, опираясь на важность каждой колонки, делает вывод.

LGBM Classifier видит и сравнивает ответы RandomForestClassifier и колонки “Churn”, которую LGBM и пытается предсказать. Так как RandomForestClassifier отвечает с точностью 100% на тренировочном датасете, независимой модели нет смысла проводить собственный анализ, и модель начинает просто перенимать ответы с RandomForestClassifier.

Появляется новая гипотеза: «*…раз RandomForestClassifier и подобные ему модели переобучаются на тренировке, давайте возьмем модели, которые не подвержены переобучению на тренировке*»

Проверка: я убрал две модели, которые переобучились на тренировке. Остались две:

```
Models (train)          Metrics (train)
----------------------  ------------------------------------------------------------------
LogisticRegression()    {'accuracy': 0.858589647411853, 'f1-score': 0.2873345935727788}
KNeighborsClassifier()  {'accuracy': 0.9246061515378845, 'f1-score': 0.6763285024154588}


Models (test)           Metrics (test)
----------------------  ------------------------------------------------------------------
LogisticRegression()    {'accuracy': 0.8740629685157422, 'f1-score': 0.3}
KNeighborsClassifier()  {'accuracy': 0.904047976011994, 'f1-score': 0.5294117647058824}
```

Добавил ответы только двух этих моделей в таблицу и обучил независимую модель LGBM Classifier на новой таблице. Результаты получились следующие:

- Accuracy: 0.9295352323838081
- F1-score: 0.7044025157232705

Точность явно повысилась, однако если обучить LGBM Classifier отдельно без других моделей, то точность LGBM Classifier получится следующей:

```
Models (test)           Metrics (test)
----------------------  ------------------------------------------------------------------
LogisticRegression()    {'accuracy': 0.8740629685157422, 'f1-score': 0.3}
KNeighborsClassifier()  {'accuracy': 0.904047976011994, 'f1-score': 0.5294117647058824}
LGBMClassifier()        {'accuracy': 0.9415292353823088, 'f1-score': 0.7547169811320754}
```

Можно увидеть, что точность LGBM Classifier, отдельно больше, причем Accuracy увеличилась на 0.02 пункта, а F1-score на 0.05 пунктов.

Заключение

Применение метамодели не увеличило точность предсказаний. Проверяя первую гипотезу, изменения были незначительны, проверяя вторую гипотезу, точность оказалась не самой оптимальной из всех возможных.