**Отчет о построении модели для задачи «Титаник»**

**Описание задачи**.

Определить вероятность выживания пассажира в случае крушения корабля Титаник на основе данных полученных во время посадки на корабль.

Работа проведена jupyter notebook.

**Описание данных.**

Взят тренировочный сет из <https://www.kaggle.com/c/titanic/data>

Он содержит в себе записи 891 пассажира с проставленной информации о том пережил ли пассажир кораблекрушение.

**Описание целевой.**

Бинарная. 1 = факт того, остался ли человек жив после кораблекрушения.

**Выбранная метрики.**

Так как я хочу больше сфокусироваться на выделении групп пассажиров с высокой или низкой вероятностью выживании, я буду придерживаться классической метрики для задач ранжирования - ROC\_AUC.

**Разбиение данных.**

Для работы данные были разделены на треин и тест. (Реальный тест с каггла не использовался, так как не имеет целевой, далее я буду называть его «*реальными*» данными)

Разбиение было сделано случайно, но стратифицировано по целевой и по полу. Так как мы уже имели информацию о том, что пол является сильным предиктором и мы не ожидаем сильного смещения в распределении этой переменной в «реальных» данных.

|  | **size** | **mean target** |
| --- | --- | --- |
| **train** | 668 | 0.383234 |
| **test** | 223 | 0.385650 |

**Первые 3 строки данных**

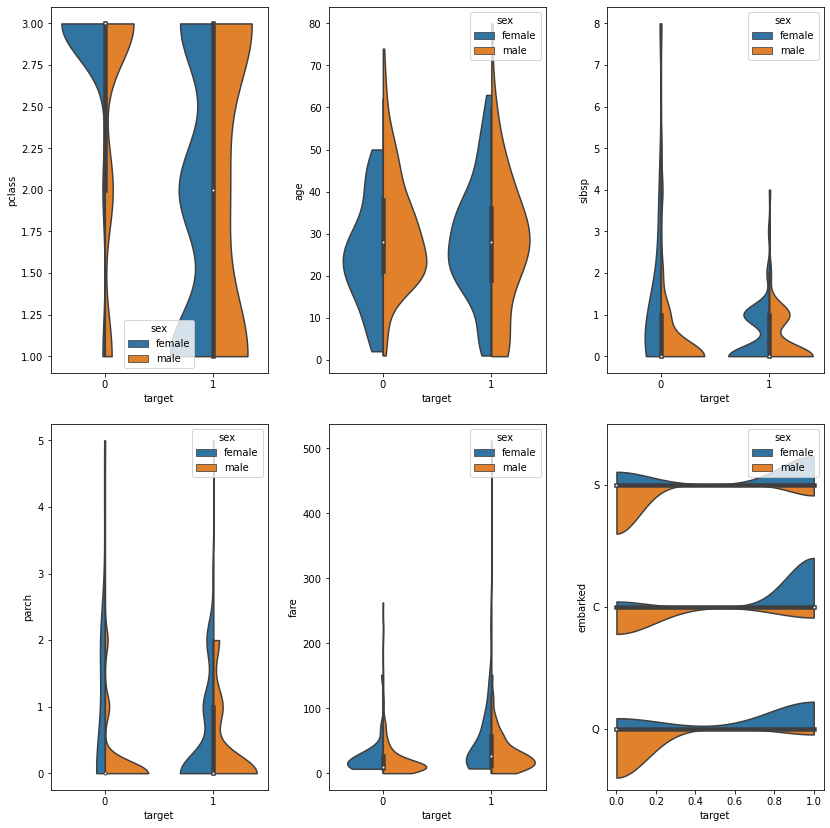
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **target** | **pclass** | **name** | **sex** | **age** | **sibsp** | **parch** | **ticket** | **fare** | **cabin** | **embarked** |
| **PassengerId** |  |  |  |  |  |  |  |  |  |  |  |
| **1** | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | июл.00 | NaN | S |
| **2** | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **3** | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | июл.50 | NaN | S |

**Визуализация исходных данных.**

**Только тренировочные данные**

На этих графиках, по оси Х происходит разбиение на целевую, а скрипки разделены по полу.

Ширина скрипок – отражает распределение переменной по значениям на оси Y.

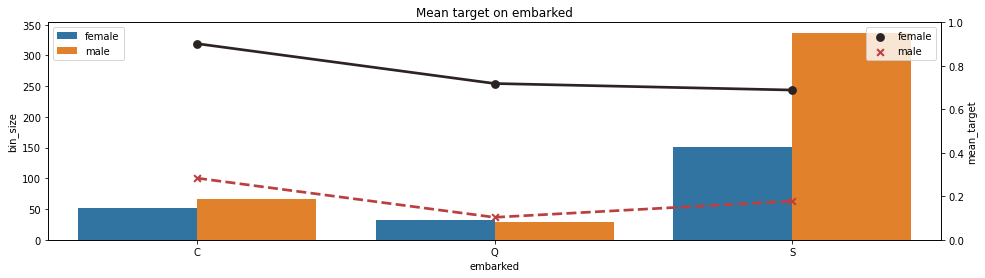
****

**Далее посмотрим на среднюю целевую в группах переменных.**

На этих графиках на оси Х показаны сгруппированные значения переменных (или их категория), на оси У слева – количество наблюдений в этой категории, а справа Средняя целевая. При этом графики разбиты по полу.

То есть, столбцы – количество м/ж в группе, а точки (линии), средняя выживаемость м/ж группы.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |



Можно заметить корреляции переменных с целевой, но этих данных недодасточно для точного прогноза. Попробуем извлечь предикторы из других переменных.

**Генерация предикторов.**

Так как в данных имеются переменные, которые затруднительно применять в моделях напрямую, мы попробуем сгенерировать из них числовые или категориальные предикторы.

Для имени выполним такие операции:

* Посчитаем количество слов
* Добавим флаги титула. (mr, miss, mrs, редкий титул и т.д.)
* Посчитаем длину и количество слов
* Самое длинное имя
* Топ-10 самых популярных имен

Для билета выведем переменные – количество цифр, наличие букв.

Для номера кабины – наличие кабины, буква кабины, количество кабин.

Также посчитаем количество родственников и выведем бинарную переменную – путешествует один.

**Дополнительная предобработка данных.**

Предобработка переменных, такие как борьба с пустыми значениями и работа с категориальными предикторами будет проходить в рамках пайплайнов для каждой модели индивидуально и будет анализироваться как гиперпараметр модели.

**Корреляционный анализ.**

Мы попарно сравнили все переменные и обнаружили, что имеются переменные с высокой корреляцией.

|  |  |  |
| --- | --- | --- |
| Корреляция Спирмана | Переменная 1 | Переменная 2 |
| 0.99710 | n\_cabins | has\_cabin |
| 0.96065 | alone | n\_relatives |

Это числовое бинарное их представление. Оставим только бинарное представление.

**Information Value.**

Проведя WOE трансормацию на данных, которые содержали сгенерированные предикторы, посчитаем IV.

|  |  |
| --- | --- |
| Переменная | IV value |
| **ticket\_has\_text** | 0.000773 |
| **has\_dr** | 0.002113 |
| **popular\_name** | 0.010190 |
| **parch** | 0.013387 |
| **has\_rev** | 0.015664 |
| **has\_master** | 0.022157 |
| **name\_length\_mean** | 0.022920 |
| **sibsp** | 0.040966 |
| **has\_rare\_title** | 0.049862 |
| **name\_longest** | 0.081308 |
| **age** | 0.115221 |
| **embarked** | 0.122596 |
| **alone** | 0.153021 |
| **name\_length** | 0.287910 |
| **ticket\_num\_len** | 0.289836 |
| **name\_size** | 0.329218 |
| **has\_cabin** | 0.435560 |
| **has\_miss** | 0.465755 |
| **fare** | 0.482095 |
| **cabin\_letter** | 0.485941 |
| **has\_mrs** | 0.486594 |
| **pclass** | 0.554747 |
| **sex\_male** | 1.330268 |
| **has\_mr** | 1.358101 |

Очевидно, что не все переменные имеют значимую предиктивную силу. Но не будем их отбрасывать сразу.

Также проведем отбор предикторов посредством внутренних алгоритмов CatBoost.

Алгоритм последовательно отбрасывает переменные и замеряет loss\_value на тестовом сете.

|  |  |
| --- | --- |
| **Отброшенный предиктор** | **Loss value Test** |
| Все предикторы | 0.376763 |
| has\_dr | 0.381161 |
| ticket\_num\_len | 0.385515 |
| ticket\_has\_text | 0.389298 |
| has\_rare\_title | 0.385540 |
| popular\_name | 0.381831 |
| has\_mrs | 0.379234 |
| alone | 0.379250 |
| name\_longest | 0.377338 |
| has\_rev | 0.376176 |
| has\_miss | 0.374829 |
| parch | 0.373385 |
| cabin\_letter | 0.371161 |
| name\_size | 0.374905 |
| name\_length\_mean | 0.378885 |
| embarked | 0.372585 |
| has\_master | 0.390056 |
| name\_length | 0.386129 |
| sibsp | 0.393811 |
| age | 0.423858 |
| has\_cabin | 0.400488 |
| fare | 0.511689 |
| pclass | 0.651568 |
| sex\_male | 0.653632 |

Отброс последних предикторов очевидно снижает предсказательную способность модели.

**Результат отбора переменных.**

Переменные 'name', 'ticket', 'cabin', были отброшены, но заменены на сгенерированные на их основе предикторы. 'sex' был заменен на бинарный sex\_male.

Предикторы с высокой корреляцией также были отброшены (их бинарные аналоги остаилсь).

*Остальные переменные разделим на группы.*

Итого получаем поднаборы переменных, которые будем сравнивать между собой.

*«Все предикторы»:*

'pclass', 'age', 'sibsp', 'parch', 'fare', 'embarked', 'name\_size', 'name\_length', 'name\_longest', 'name\_length\_mean', 'popular\_name', 'has\_mr ', 'has\_miss ', 'has\_mrs ', 'has\_master ', 'has\_rev ', 'has\_dr ', 'has\_rare\_title', 'ticket\_num\_len', 'ticket\_has\_text', 'has\_cabin', 'cabin\_letter', 'alone', 'sex\_male'

*«Потенциально хорошие предикторы»:*

'parch', 'name\_size', 'name\_length\_mean', 'embarked', 'name\_length', 'cabin\_letter', 'has\_master ', 'sibsp', 'age', 'fare', 'has\_cabin', 'pclass', 'sex\_male', 'has\_mr '

*«Хорошие предикторы»:*

'name\_length', 'sibsp', 'age', 'has\_cabin', 'fare', 'pclass', 'sex\_male', 'has\_mr '

**Обучение и сравнение моделей.**

Для сравнения моделей и их параметров будем применять кросс валидацию. Но для большей уверенности в результате оценки, разбиение будем делать 10 раз (случайным образом, но одинаковым для каждой модели) по 5 фолдов. В результате мы будем получать 50 скоров на разных отложенных фолдах. Сравнивать будем не только среднее значение, но будем применять Т тест, на то что скоры принадлежат разным распределениям. То есть будем сравнивать то, насколько каждая из моделей лучше или хуже другой на соответствующем фолде.

**Baseline\_0.**

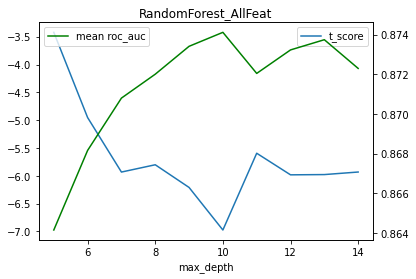
Сначала обучим базовую модель.

Возьмем RandomForestClassifier, пустые значения заполним most\_frequent. Переменные –

'pclass', 'sex\_male', 'age', 'sibsp', 'fare'

**Сравнение 1.**

Сравнивать будем с RandomForestClassifier но ужа на всех предикторах. Также будем менять параметр max\_depth.



Можно заметить, что Т статистика оказалось наименьшей при параметре равном 10 и среднее значение скора на фолдах, также максимально. Возьмем эту модель как новый бейслан и будем продолжать эксперименты.

**Далее, таким же образом было проведено сравнение других моделей.**

WOE трансформацию + LogisticRegression.

Были проверены разные наборы предикторов и разные варианты биннинга WOE.

Аналогично продолжили с CatBoostClassifier, выбирая разные значения параметра iterations.

В связи с недостатком времени, было проведено лишь несколько экспериментов.

Итоговые результаты были сведены в таблицу.

**Итоговые результаты. Метрика – ROC\_AUC.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **RandomForest(max\_depth=10) + SimpleImputer(numeric: mean, categorical: most\_frequent) + OneHotEncoder** | | |
|  | all\_features | potential\_good\_features | good\_features |
| train | 0.9905 | 0.9874 | 0.9860 |
| test | 0.8934 | 0.8939 | 0.8953 |
|  | **WOE (max\_groups=4, missing as separate group) + LogisticRegression** | | |
|  | all\_features | potential\_good\_features | good\_features |
| train | 0.8764 | 0.8723 | 0.8593 |
| test | 0.8885 | 0.8858 | 0.8659 |
|  | **CatBoostClassifier(iterations=200)** | | |
|  | all\_features | potential\_good\_features | good\_features |
| train | 0.9545 | 0.9413 | 0.9485 |
| test | 0.9026 | 0.9015 | 0.9010 |
|  | **VotingClassifier('soft'). Combination of those 3** | | |
|  | all\_features | potential\_good\_features | good\_features |
| train | 0.9563 | 0.9488 | 0.9502 |
| test | 0.9015 | 0.8981 | 0.8947 |

**Выводы:**

Модели ансамблей показывают более высокие скоры, чем линейная модель на WOE трансформации, но для бизнес целей обоснование прогноза таких моделей, может оказаться затруднительным.

Комбинация полученных моделей, не привела к росту результатов.

**Следующие шаги:**

Стоит рассмотреть возможность создания комбинаций переменный и расширения списка предикторов. После этого провести более тщательный отбор переменных и расширить процесс подбора параметров моделей.

Далее рекомендуется провести анализ ошибок моделей кандидатов и заняться их интерпретацией (например путем shap).

Небольшая демонтрация по анализу ошибок через shap, расположена в конце второго джупитера.