

Содержание

Об авторе	11
Колофон	11
Введение	13
Чего ожидать	14
Для кого написана эта книга	14
Соглашения, принятые в этой книге	14
Использование примеров кода	15
Посвящение	16
Ждем ваших отзывов!	17
Глава 1. Введение в машинное обучение	19
Установка с использованием pip	23
Установка с помощью conda	24
Глава 2. Обзор процесса машинного обучения	27
Глава 3. Пошаговая классификация: набор данных Titanic	29
Соображения о плане проекта	29
Импорт	30
Задать вопрос	31
Условия для данных	31
Сбор данных	33
Очистка данных	34
Создание признаков	41
Выборка данных	43
Замещение данных	44
Нормализация данных	45
Рефакторинг	46

Простая модель	47
Разные семейства	48
Стекирование	49
Создание модели	50
Оценка модели	51
Оптимизация модели	52
Матрица неточностей	53
Кривая ROC	55
Кривая обучения	56
Развертывание модели	57
Глава 4. Пропущенные данные	59
Изучение пропущенных данных	60
Отбрасывание пропущенных данных	63
Замещение данных	64
Добавление индикаторных столбцов	65
Глава 5. Очистка данных	67
Имена столбцов	67
Замена пропущенных значений	68
Глава 6. Исследование	71
Размер данных	71
Сводная статистика	72
Гистограмма	73
Диаграмма рассеяния	74
Объединенный график	75
Парная сетка	77
Диаграмма размаха и скрипичная диаграмма размаха	78
Сравнение двух порядковых значений	80
Корреляция	82
RadViz	86
Параллельные координаты	88
Глава 7. Предварительная обработка данных	91
Стандартизация	91
Масштабирование до диапазона	93
Фиктивные переменные	94
Меточное кодирование	95

Частотное кодирование	96
Извлечение категорий из строк	97
Другие категориальные кодирования	99
Конструирование признаков данных	101
Добавление признака col_на	102
Конструирование признаков вручную	103
Глава 8. Выбор признаков	105
Коллинеарные столбцы	106
Регрессия лассо	108
Удаление рекурсивных признаков	110
Взаимная информация	112
Анализ основных компонентов	112
Важность признака	113
Глава 9. Несбалансированные классы	115
Использование другой метрики	115
Алгоритмы и ансамбли на основе дерева	115
Штрафующие модели	116
Повышающая дискретизация миноритарного класса	116
Генерация данных миноритарного класса	117
Понижающая дискретизация мажоритарного класса	118
Повышающая дискретизация, затем понижающая	120
Глава 10. Классификация	121
Логистическая регрессия	122
Наивный байесовский классификатор	127
Метод опорных векторов	129
K-ближайшие соседи	133
Дерево решений	136
Случайный лес	143
XGBoost	149
Градиентный бустинг с LightGBM	159
TROT	165
Глава 11. Выбор модели	169
Кривая валидации	169
Кривая обучения	171

Глава 12. Метрики и оценка классификации	173
Матрица неточностей	173
Метрики	176
Корректность	178
Отзыв	178
Точность	179
F1	179
Отчет о классификации	179
ROC	180
Кривая “точность—отзыв”	181
График кумулятивного усиления	183
Кривая подъема	185
Баланс классов	186
Ошибка прогнозирования класса	187
Порог дискриминации	188
Глава 13. Объяснение моделей	191
Коэффициенты регрессии	191
Важность признака	192
LIME	192
Интерпретация дерева	194
Графики частичной зависимости	195
Суррогатные модели	198
Sharpley	199
Глава 14. Регрессия	205
Базовая модель	207
Линейная регрессия	208
SVM	212
K-ближайшие соседи	214
Древо решений	216
Случайный лес	223
Регрессия XGBoost	226
Регрессия LightGBM	232
Глава 15. Метрики и регрессионная оценка	239
Метрики	239
График остатков	242

Гетероскедастичность	242
Нормальные остатки	244
График ошибки прогноза	245
Глава 16. Объяснение регрессионных моделей	247
Sharpley	247
Глава 17. Уменьшение размерности	253
PCA	253
UMAP	271
t-SNE	277
PHATE	281
Глава 18. Кластеризация	285
Метод k-средних	285
Агломерационная (иерархическая) кластеризация	292
Понятие кластеров	295
Глава 19. Конвейер	301
Классификационный конвейер	301
Конвейер регрессии	303
Конвейер PCA	304
Предметный указатель	307

Выбор признаков

Мы используем *выбор признаков* (feature selection) для отбора тех признаков, которые полезны для модели. Нерелевантные признаки могут оказать негативное влияние на модель. Коррелированные признаки могут сделать коэффициенты регрессии (или важность признаков в древовидных моделях) нестабильными или трудными для интерпретации.

Проклятие размерности (curse of dimensionality) — это еще одна проблема, которую стоит рассмотреть. По мере увеличения количества размерностей ваших данных они становятся все более и более разреженными. Это может затруднить получение сигнала, если у вас нет больше данных. По мере добавления размерностей вычисления соседей, как правило, теряют свою полезность.

Кроме того, время обучения обычно зависит от количества столбцов (и иногда оно даже хуже линейного). Обеспечив краткость и точность своих столбцов, вы можете получить лучшую модель за меньшее время. Мы рассмотрим несколько примеров, используя набор данных `agg_df` из предыдущей главы. Помните, что это набор данных Titanic с некоторыми дополнительными столбцами с информацией о каюте. Поскольку этот набор данных агрегирует числовые значения для каждой каюты, он покажет много корреляций. К другим вариантам относятся PCA и поиск древовидного классификатора `.feature_importances_`.

Коллинеарные столбцы

Чтобы найти столбцы с коэффициентом корреляции 0,95 или выше, можно использовать определенный ранее признак `correlated_columns` или запустить следующий код:

```
>>> limit = 0.95
>>> corr = agg_df.corr()
>>> mask = np.triu(
...     np.ones(corr.shape), k=1
... ).astype(bool)
>>> corr_no_diag = corr.where(mask)
>>> coll = [
...     c
...     for c in corr_no_diag.columns
...     if any(abs(corr_no_diag[c]) > threshold)
... ]
>>> coll
['pclass_min', 'pclass_max', 'pclass_mean',
 'sibsp_mean', 'parch_mean', 'fare_mean',
 'body_max', 'body_mean', 'sex_male', 'embarked_S']
```

Показанный ранее визуализатор Rank2 библиотеки Yellowbrick построит тепловую карту корреляций.

Пакет `rfpimp` способен визуализировать *мультиколлинеарность* (multicollinearity). Функция `plot_dependence_heatmap` обучает случайный лес по каждому числовому столбцу из других столбцов в обучающем наборе данных. Значение зависимости — это оценка R^2 из *оценки OOB* (Out Of Bag — не вошедший в набор) для прогнозирования этого столбца (рис. 8.1).

Предлагаемый способ использования этого графика — найти значения, близкие к 1. Метка на оси X — это признак, который прогнозирует метку на оси Y. Если признак прогнозирует другую метку, вы можете удалить спрогнозированный признак (признак по оси Y). В нашем примере `fare` прогнозирует `pclass`, `sibsp`, `parch` и `embarked_Q`. Мы должны сохранить `fare` и, удалив другие признаки, получить аналогичные показатели:

```

>>> rfpimp.plot_dependence_heatmap(
...     rfpimp.feature_dependence_matrix(X_train),
...     value_fontsize=12,
...     label_fontsize=14,
...     figsize=(8, 8),sn
... )
>>> fig = plt.gcf()
>>> fig.savefig(
...     "images/mlpr_0801.png",
...     dpi=300,
...     bbox_inches="tight",
... )

```

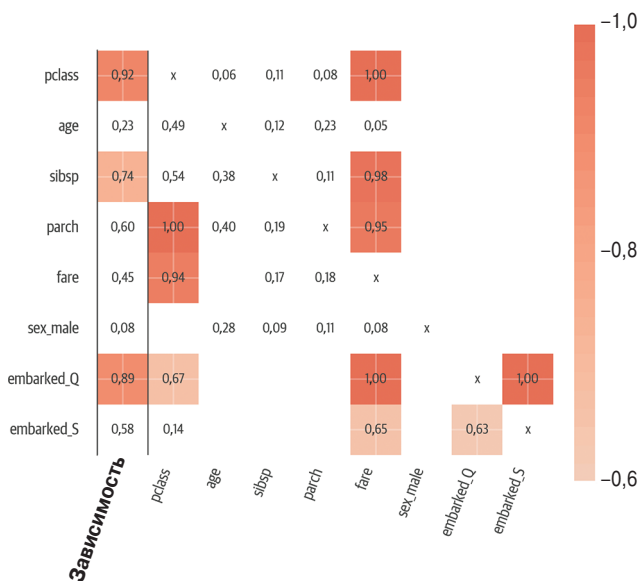


Рис. 8.1. Тепловая карта зависимости. Признаки *pclass*, *sibsp*, *parch* и *embarked_Q* можно прогнозировать из *fare*, поэтому мы можем удалить их

Вот код, показывающий, что мы получим похожую оценку, если уберем эти столбцы:

```

>>> cols_to_remove = [
...     "pclass",

```



```

...     "sibsp",
...     "parch",
...     "embarked_Q",
... ]
>>> rf3 = RandomForestClassifier(random_state=42)
>>> rf3.fit(
...     X_train[
...         [
...             c
...             for c in X_train.columns
...             if c not in cols_to_remove
...         ]
...     ],
...     y_train,
... )
>>> rf3.score(
...     X_test[
...         [
...             c
...             for c in X_train.columns
...             if c not in cols_to_remove
...         ]
...     ],
...     y_test,
... )
0.7684478371501272

>>> rf4 = RandomForestClassifier(random_state=42)
>>> rf4.fit(X_train, y_train)
>>> rf4.score(X_test, y_test)
0.7659033078880407

```

Регрессия лассо

Если вы используете регрессию лассо¹, можете установить альфа-параметр, который действует, как параметр регуляризации. Когда вы увеличиваете его значение, оно придает меньший вес менее важным признакам. Здесь мы используем

¹ Least Absolute Shrinkage and Selection Operator — LASSO. — *Примеч. ред.*

модель LassoLarsCV для итерации по различным значениям альфа и отслеживания коэффициентов признаков (рис. 8.2):

```
>>> from sklearn import linear_model
>>> model = linear_model.LassoLarsCV(
...     cv=10, max_n_alphas=10
... ).fit(X_train, y_train)
>>> fig, ax = plt.subplots(figsize=(12, 8))
>>> cm = iter(
...     plt.get_cmap("tab20")(
...         np.linspace(0, 1, X.shape[1])
...     )
... )
>>> for i in range(X.shape[1]):
...     c = next(cm)
...     ax.plot(
...         model.alphas_,
...         model.coef_path_.T[:, i],
...         c=c,
...         alpha=0.8,
...         label=X.columns[i],
...     )
>>> ax.axvline(
...     model.alpha_,
...     linestyle="--",
...     c="k",
...     label="alphaCV",
... )
>>> plt.ylabel("Regression Coefficients")
>>> ax.legend(X.columns, bbox_to_anchor=(1, 1))
>>> plt.xlabel("alpha")
>>> plt.title(
...     "Regression Coefficients Progression for Lasso
Paths"
... )
>>> fig.savefig(
...     "images/mlpr_0802.png",
...     dpi=300,
...     bbox_inches="tight",
... )
```

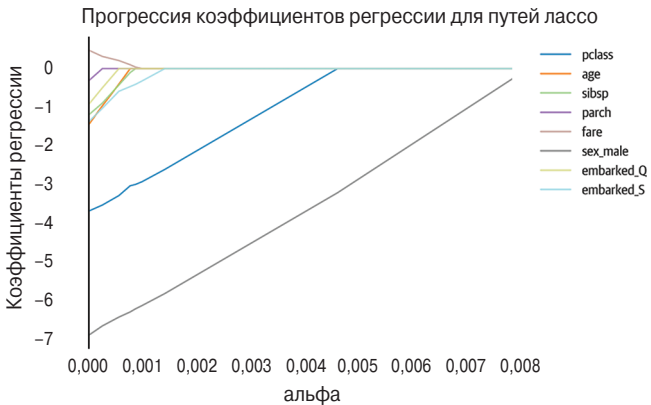


Рис. 8.2. Коэффициенты признаков при варьировании альфа во время регрессии Лассо

Удаление рекурсивных признаков

При удалении рекурсивных признаков удаляются самые слабые признаки, а затем модель подгоняется (рис. 8.3). Для этого модель передается Scikit-learn с атрибутом `.coef_` или `.feature_importances_`:

```
>>> from yellowbrick.features import RFECV
>>> fig, ax = plt.subplots(figsize=(6, 4))
>>> rfe = RFECV(
...     ensemble.RandomForestClassifier(
...         n_estimators=100
...     ),
...     cv=5,
... )
>>> rfe.fit(X, y)

>>> rfe.rfe_estimator_.ranking_
array([1, 1, 2, 3, 1, 1, 5, 4])

>>> rfe.rfe_estimator_.n_features_
4
>>> rfe.rfe_estimator_.support_
```

```
array([ True,  True, False, False,  True,
        True, False, False])
```

```
>>> rfe.poof()
>>> fig.savefig("images/mlpr_0803.png", dpi=300)
```

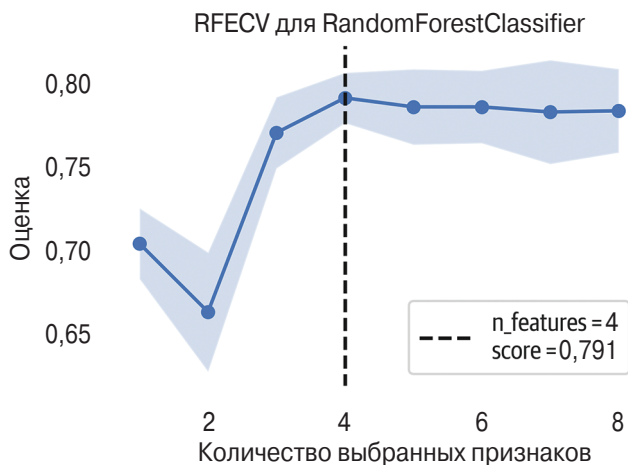


Рис. 8.3. Удаление рекурсивных признаков

Мы будем использовать удаление рекурсивных признаков, чтобы найти 10 наиболее важных признаков. (В этом агрегированном наборе данных мы обнаруживаем утечку в столбце survival!)

```
>>> from sklearn.feature_selection import RFE
>>> model = ensemble.RandomForestClassifier(
...     n_estimators=100
... )
>>> rfe = RFE(model, 4)
>>> rfe.fit(X, y)
>>> agg_X.columns[rfe.support_]
Index(['pclass', 'age', 'fare', 'sex_male'],
      dtype='object')
```

Взаимная информация

Библиотека Scikit-learn предоставляет непараметрические тесты, использующие метод *k*-ближайшего соседа для определения *взаимной информации* (mutual information) между признаками и целью. Взаимная информация определяет количество информации, полученной при наблюдении за другой переменной. Значение равно нулю или больше. Если значение равно нулю, то никакой связи между ними нет (рис. 8.4). Это число не ограничено и представляет количество *битов* (bit), совместно используемых признаком и целью:

```
>>> from sklearn import feature_selection

>>> mic = feature_selection.mutual_info_classif(
...     X, y
... )
>>> fig, ax = plt.subplots(figsize=(10, 8))
>>> (
...     pd.DataFrame(
...         {"feature": X.columns, "vimp": mic}
...     )
...     .set_index("feature")
...     .plot.barh(ax=ax)
... )
>>> fig.savefig("images/mlpr_0804.png")
```

Анализ основных компонентов

Другой вариант выбора признака — запуск *анализа основных компонентов* (principal component analysis). Если у вас есть главные основные компоненты (main principal component), изучите признаки, которые вносят наибольший вклад. Это признаки, которые имеют наибольшую вариацию. Обратите внимание, что это алгоритм без учителя, который не учитывает *y*.

Более подробная информация по этой теме приведена далее в книге.

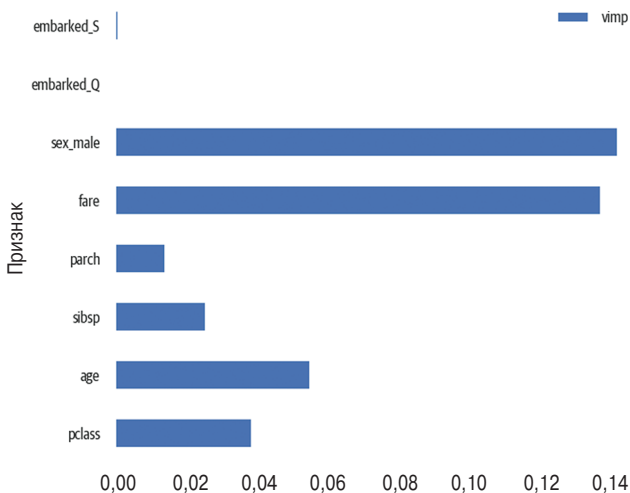


Рис. 8.4. Диаграмма взаимной информации

Важность признака

После обучения большинство древовидных моделей предоставляют доступ к атрибуту `.feature_importances_`. Более высокая важность обычно означает, что при удалении признака из модели возникает большая ошибка. Более подробная информация о древовидных моделях приведена в соответствующих главах.