Математическое обоснование к проекту

Разработка алгоритмов машинного обучения для прогнозирования поведения киберфизической системы

Выполнил:

студент Гавриленко О. Р

Санкт-Петербург

2024

Оглавление

Глава 1

Описание проблемы

Установки скважинных штанговых насосов широко применяются для эксплуатации скважин разных категорий на различных нефтяных месторождениях страны. Одной из основных задач проектирования эксплуатации скважин СШНУ является обоснование некоторых технологических характеристик, среди которых: дебит, забойное давление, давление на приеме насоса, глубина спуска насоса. Дебит и забойное давление зависят от принятой системы разработки, ее текущего состояния, а также от объективных ограничений.

Расчет давления на приеме насоса представляет сложную технико-экономическую задачу, решение которой связано с определенными допущениями. Существуют различные рекомендации о величине давления на приеме насоса, связанные с газовым фактором и обводненностью продукции.

Обычно расчеты этой характеристики делаются путем математического моделирования физических законов [1], однако для конкретного численного решения на модель накладываются определенные ограничения.

Также для расчетов применяется корреляционный анализ [2], но машинное обучение за счет более гибких моделей и числа параметров может дать лучшие результаты.

В данной работе исследуется эффективность различных современных регрессионных алгоритмов машинного обучения на открытых данных ПАО "Газпром".

Глава 2

Техническое описание решения

2.1. Первоначальная работа с данными

2.1.1. Информация о данных

Данные были взяты с соревнования Яндекса (ссылка на данные). Они представляют собой 4000 строк, каждая из которых состоит из следующих признаков:

1. Диаметр экспл. колонны
2. Буферное давление
3. Динамическая высота
4. Затрубное давление
5. Обводненность
6. Глубина спуска
7. Давление на приёме

После импорта данные были разделены на тренировочную, валидационную и тестовую выборку. На тренировочной выборке обучалась первоначальная модель, затем гиперпараметры (если они были у модели) подбирались на валидационной выборке, затем итоговая модель исследовалась на тестовой выборке и сравнивалась с другими.

По совместным распределениям и корреляциям можно сказать, что есть зависимые сильнокоррелированные признаки (буферное давление/давление в линии, буферное давление/затрубное давление, затрубное давление/давление в линии). Это означает, что скорее всего для некоторых моделей будет нужна процедура регуляризации.

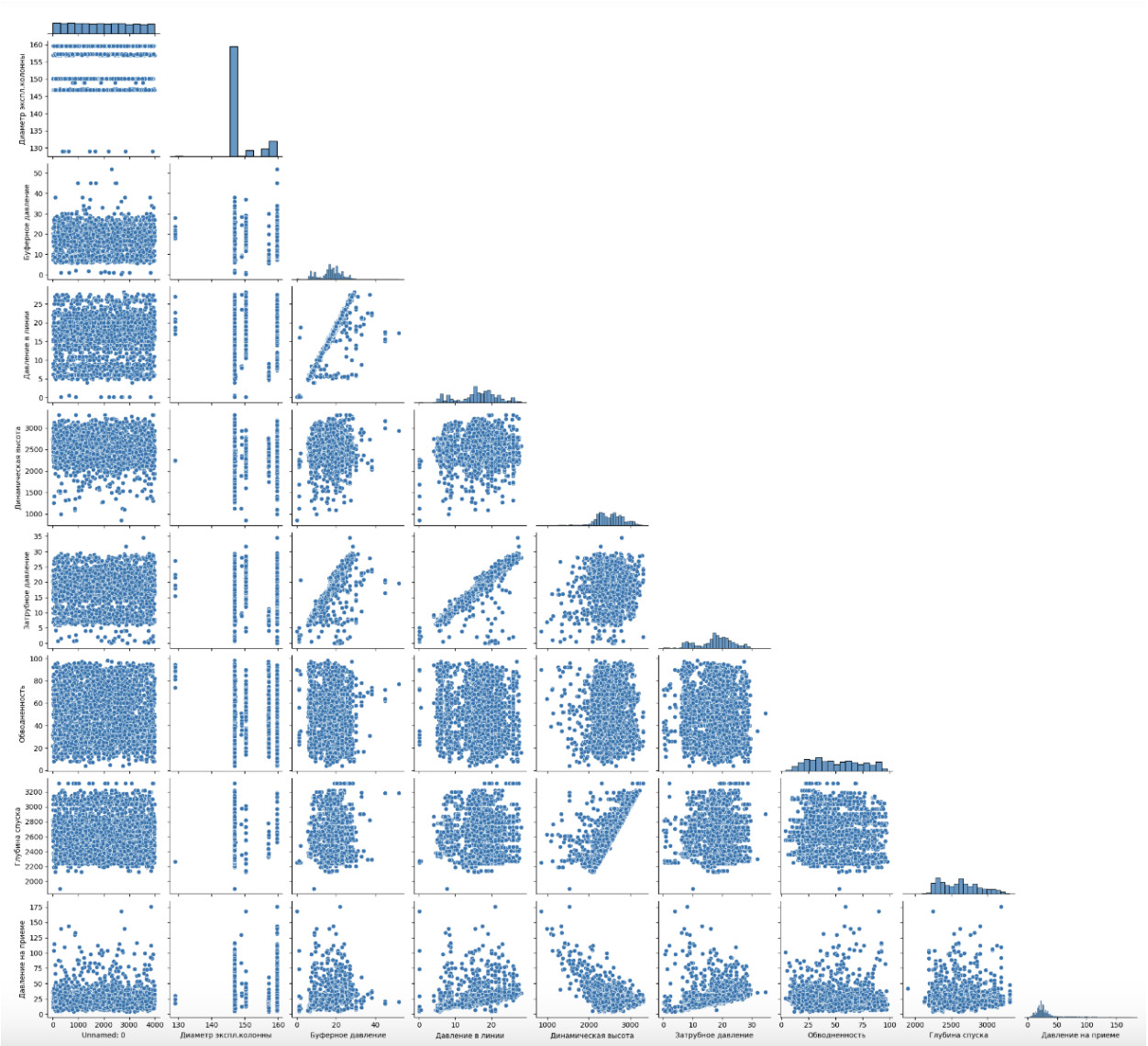

Рис. 2.1. График совместных распределений

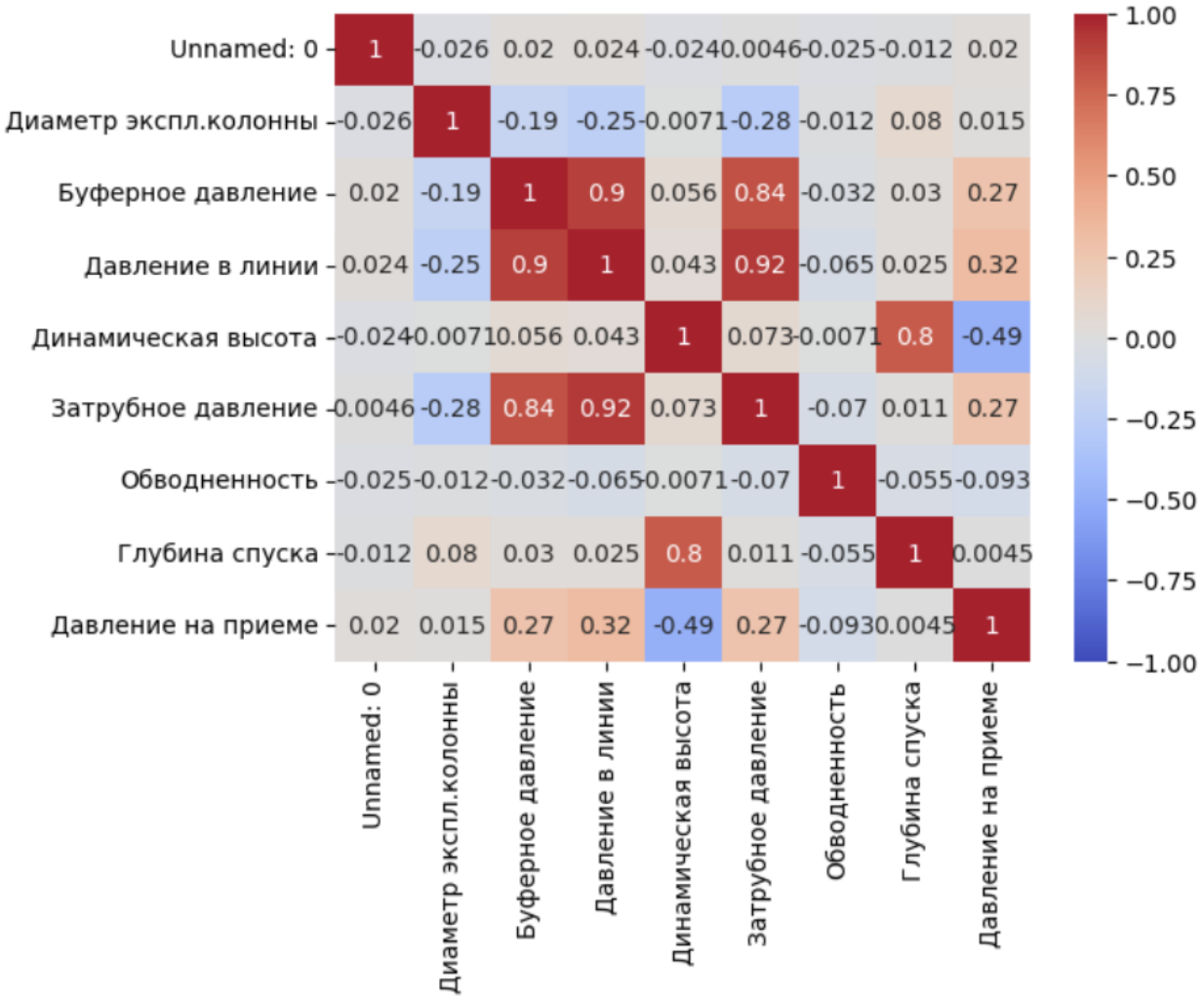

Рис. 2.2. График совместных корреляций

2.2. Применение моделей машинного обучения

2.2.1. О метрике качества

Коэффициент детерминации (R^2) — это доля дисперсии зависимой переменной, объясняемая рассматриваемой моделью зависимости, то есть объясняющими переменными.

$$R^2 = 1 - \frac{\hat{\sigma}^2}{\hat{\sigma}_y^2} = 1 - \frac{SS_{res}/n}{SS_{tot}/n} = 1 - \frac{SS_{res}}{SS_{tot}}, \tag{2.1}$$

где $SS_{res} = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ — сумма квадратов остатков регрессии, y_i, $\hat{y}_i$ —

фактические и расчётные значения объясняемой переменной. $SS_{tot} = \sum_{i=1}^{n}(y_i - \overline{y})^2 = n\hat{\sigma}_y^2$ — общая сумма квадратов, $\overline{y}$ — среднее значение фактического значения объясняемой переменной.

Эта метрика является распространенной для оценки качества регрессионных моделей. Чем ближе ее значение к 1, тем слабее отличается предсказанный результат от реального.

2.2.2. Линейная регрессия

Пусть имеем матрицу данных $\mathbf{X}$ состоящую из n индивидов $\mathbf{x}_i \in \mathbb{R}^p$. Предполагаем, что имеем функциональную завимость между $\mathbf{X}$ и вектором ответов $\mathbf{y}$, состоящего из y_i:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \qquad i = 1, \ldots, n. \tag{2.2}$$

Где ε_i — ошибка обучения или остаток. Предполагаем что они независимые, что $\mathbb{E}\varepsilon = 0$, и $\mathbb{D}\varepsilon = \sigma^2$.

Задача: найти $\hat{f}$ (оценку функции регрессии f), чтобы для каждого индивида $y_i \approx \hat{f}(\mathbf{x}_i)$. В модели линейной регрессии f представляет собой линейную функцию.

К недостаткам линейной регрессии относят:

1. Выбросы оказывают большое влияние на итоговую функцию
2. Предполагается, что зависимость между признаками линейная. В других случаях линейная регрессия работает плохо
3. Легко переобучается при сильнокоррелируемых признаках

Метрика R^2 для линейной регрессии на наших данных показала слабую работу модели, что было ожидаемо, так как зависимость давления на приеме от других признаков не похожа на линейную.

2.2.3. Решающие деревья

Пусть $\mathbf{X} \in \mathbb{R}^{n \times k}$ — матрица данных, n — число индивидов, k — число признаков, $Y \in \mathbb{R}^n$ — вектор отклика, X_i — вектор признака, $i \in 1 : k$.

Модель 1. Линейная регрессия со всеми признаками

```
selected_features = ['Диаметр экспл.колонны', 'Буферное давление', 'Давление в линии', 'Динамическая высота', 'Затру
x = df_train[selected_features]
y = df_train['Давление на приеме']
best_model_reg = LinearRegression().fit(x, y)
y_predict = best_model_reg.predict(x)

print("Train R^2:", r2_score(y_predict, y.to_numpy()))
x = df_valid[selected_features]
y = df_valid['Давление на приеме']
y_predict = best_model_reg.predict(x)
print()
print("Valid R^2:", r2_score(y_predict, y.to_numpy()))
```

```
Train R^2: 0.742346833907968

Valid R^2: 0.6627360568209681
```

Рис. 2.3. Результат линейной регрессии на тренировочной и валидационной выборке

Дерево решений — бинарное дерево (V, E), в котором каждой $v \in V$, не являющейся листом соответствует функция, являющаяся предикатом $\beta_v : X \to \{0, 1\}$, а каждому листу $l \in V$ соответствует прогноз $f_l : X \to Y$.

Деревья решений можно применять для классификации и регрессии.

Алгоритм построения дерева:

1. Выбираем признак X_j и порог s так, чтобы разбиение X^n на $R_1(j, s) = \{x \in X^n | X_j < s\}$ и $R_2(j, s) = \{x \in X^n | X_j \geq s\}$ решало оптимизационную задачу, соответствующую задаче классификации или регрессии:

$$\sum_{i:x_i \in R_1(j,s)} \text{error}(y_i, \hat{y}_{R_1}) + \sum_{i:x_i \in R_2(j,s)} \text{error}(y_i, \hat{y}_{R_2}) \to \min_{j,s},$$

   где error — функция потерь, в случае регрессии сумма остатков

2. Разбиваем выборку на области R_1 и R_2, образуя две дочерние вершины.

3. Повторяем процедуру в пределах каждой получаемой области, пока не выполнится критерий остановки.

Очевидный критерий остановки — остановка в случае если все индивиды в листе принадлежат одному классу. Дополнительно можно задать следующие критерии остановки:

1. Ограничение максимальной глубины дерева

2. Ограничение минимального числа объектов в поддереве

3. Информативность равна или меньше определенного значения

4. Ограничение максимального числа индивидов в листе

К недостаткам решающих деревьев относят:

1. Решение является лишь локально оптимальным в каждом узле и может быть неоптимальным для всего дерева

2. Метод явялется неустойчивым и склонным к переобучению

В нашем случае решающие деревья показали значительно более лучший результат, чем линейная регрессия.

Модель 2. Случайный лес

```
x = df_train[selected_features]
y = df_train['Давление на приеме']
best_model_rf = RandomForestRegressor(max_depth=100, random_state=0).fit(x, y)
y_predict = best_model_rf.predict(x)

print("Train R^2:", r2_score(y_predict, y.to_numpy()))
x = df_valid[selected_features]
y = df_valid['Давление на приеме']
y_predict = best_model_rf.predict(x)
print()
print("Valid R^2:", r2_score(y_predict, y.to_numpy()))
```

```
Train R^2: 0.9734488060377428

Valid R^2: 0.7856239830426661
```

Рис. 2.4. Результат решающих деревьев на тренировочной и валидационной выборке

2.2.4. Градиентный бустинг

Предыдущий алгоритм решающих деревьев использует подход к построению композиций, которые независимо обучают каждый базовый алгоритм по некоторому подмножеству обучающих данных. При этом возникает ощущение, что мы используем возможности объединения алгоритмов не в полную силу, и можно было бы строить их так, чтобы каждая следующая модель исправляла ошибки предыдущих. Ниже мы рассмотрим метод, который реализует эту идею — градиентный бустинг. Он работает для любых дифференцируемых функций потерь и является одним из наиболее мощных и универсальных на сегодняшний день.

Рассмотрим задачу минимизации квадратичного функционала:
$\frac{1}{2}\sum_{i=1}^{l}(a(x_i) - y_i)^2 \to \min\limits_{a}$
Будем искать итоговый алгоритм в виде суммы базовых моделей $b_n(x)$:
$a_N(x) = \sum_{n=1}^{N} b_n(x)$, где базовые алгоритмы b_n принадлежат некоторому семейству **A**.

Первый базовый алгоритм: $b_1(x) := \underset{b \in \mathbf{A}}{argmin} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - y_i)^2$.

Решение такой задачи не представляет трудностей для многих семейств алгоритмов. Теперь мы можем посчитать остатки на каждом обхекте — растояния от ответа нашего алгоритма до истинного ответа: $s_i^{(1)} = y_i - b_i(x)$

Если прибавить эти остатки к ответам построенного алгоритма, то он не будет допускать ошибок на обучающей выборке. Значит, будет разумно построить второй алгоритм так, чтобы его ответы были как можно ближе к остаткам:

$b_2(x) := \underset{b \in \mathbf{A}}{argmin} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - s^{(1)})^2$

Таким образом, каждый следующий алгоритм тоже будем настраивать на остатки предыдущих:

$s_i^{(N)} = y_i - \sum_{n=1} N - 1 b_n(x_i) = y_i - a_{N-1}(x_i), \qquad i = 1, \ldots, l$

$b_N(x) := \underset{b \in \mathbf{A}}{argmin} \frac{1}{2} \sum_{i=1}^{l} (b(x_i) - s_i^{(N)})^2$

Также, остатки могут быть найдены как антиградиент функции потерь по ответу модели, посчитанный в точке ответа уже построенной композиции:

$$s_i^{(N)} = y_i - a_{N-1}(x_i) = -\frac{\partial}{\partial z} \frac{1}{2} (z - y_i)^2 \Big|_{z = a_{N-1}(x_i)}$$

Получается, что выбирается такой базовый алгоритм, который как можно сильнее уменьшит ошибку композиции — это свойство вытекает из его близости к антиградиенту функционала на обучающей выборке.

На наших данных бустинг получил больший R^2 на тренировочной выборке чем решающие деревья, но меньший $R*^2$ на валидационной выборке, второе для нас важнее, так как мы хотим чтобы модель хорошо работала на новых данных.

Модель 3. Градиентный бустинг

```
x = df_train[selected_features]
y = df_train['Давление на приеме']
best_model_gb = GradientBoostingRegressor(random_state=0).fit(x, y)
y_predict = best_model_gb.predict(x)

print("Train R^2:", r2_score(y_predict, y.to_numpy()))
x = df_valid[selected_features]
y = df_valid['Давление на приеме']
y_predict = best_model_gb.predict(x)
print()
print("Valid R^2:", r2_score(y_predict, y.to_numpy()))
```

```
Train R^2: 0.8939835875549639

Valid R^2: 0.7962279888087785
```

Рис. 2.5. Результат бустинга на тренировочной и валидационной выборке

Список литературы

1. Лекомцев А. В., Мордвинов В. А. Определение давления у приема электроцентробежных насосов по данным исследований скважин // Недропользование. 2012. №4. URL:https://cyberleninka.ru/article/n/opredelenie-davleniya-u-priema-elektrotsentrobezhnyh-nasosov-po-dannym-issledovaniy-skvazhin (дата обращения: 10.03.2024).
2. Ponomareva I., Galkin V., Rastegaev A. (2021). Developing multilevel statistical models of determining bottom-hole flowing pressure in commercial oil well operations. Revista Ingenieria UC. 28. P. 97-110.