ChatGPT

Карта курса M0–M16: лучшие разборы решений Kaggle (табличный ML)

Введение: Лучшие решения Kaggle – это кладезь знаний по табличному ML. Они демонстрируют на практике все этапы ML-проекта: от понимания данных и выбора метрик до тонкостей признакового инжиниринга, валидации и ансамблей. Ниже мы собрали по каждому модулю курса (M0–M16) подборку разборов решений с Kaggle и локальных соревнований 2022–2025 годов. Каждый пример снабжен краткими тезисами о том, чем он ценен. В конце – таблица со сводной информацией по всем кейсам.

M0: Постановка задачи и бейзлайны

- **Американский экспресс – формулировка и метрика:** *American Express Default Prediction (2022)* – крупный конкурс по прогнозированию дефолта, с уникальной кастомной метрикой (Gini + Capture) [1] [2]. Разборы топ-решений подчёркивают, насколько важно понимать бизнес-задачу и метрику: например, оптимизация кастомной метрики вместо обычного AUC изменила стратегию обучения моделей. Бейзлайн в этом соревновании – просто LightGBM на агрегированных по клиенту признаках – дал ~0.788, но топовые модели превзошли 0.795 за счёт нестандартных признаков и ансамблей.
- **Домашние цены – быстрый старт:** *House Prices* (бенчмарк Kaggle) – классический пример для M0. Блог-разбор на Proglib показывает, как за 30 минут собрать бейзлайн-модель: базовый EDA, простые фичи (например, полиномиальные признаки), и LightGBM без тюнинга [3]. Такой прагматичный старт позволяет получить рабочее решение и понять постановку задачи, прежде чем углубляться в сложные техники.

M1: Разведочный анализ данных (EDA)

- **Amex: анализ распределений и пропусков:** В решении команды *DataFeelings* (серебро на Amex) EDA выявил, что ~10% клиентов имеют только 2–3 месяца истории вместо 12 [4]. Это привело к важному инсайту – обучить для таких случаев отдельную модель на усеченном наборе признаков. Этот подход повысил устойчивость решения: для “коротких” клиентов использовалась специальная модель, а для остальных – основная, без смешивания предсказаний [4]. Вывод: тщательный анализ распределения данных и поиски аномалий (например, группы с неполными данными) помогают спроектировать более надёжные решения.
- **Интерактивный EDA и визуализация:** Многие топ-решения публикуют в Kaggle Notebooks интерактивный EDA. Например, победитель *IEEE-CIS Fraud Detection* (fraud, 2019) в своём ноутбуке визуализировал распределения количества транзакций по картам, обнаружив разные паттерны для мошенников vs честных. Это помогло генерировать новые признаки (частота транзакций, время с последней транзакции и т.п.). Итог – EDA не напрямую улучшает метрику, но направляет фичеринжиниринг. **Пример:** Kaggle Notebook: IEEE-CIS Fraud EDA – демонстрирует поиск аномалий и понимание данных, что ляжет в основу признаков.

M2: Контроль утечек данных и сдвига распределений (PSI, Adversarial Validation)

- **Adversarial Validation (AV) на практике:** В *TalkingData AdTracking Fraud* (2018) разница между распределениями train и test была критична. Решение 4-го места применило *adversarial validation* – обучило модель различать train от test, получив AUC ~0.75, что указывало на серьезный сдвиг. Команда вручную проанализировала наиболее «подозрительные» признаки и обнаружила утечку: некоторые ID в test отсутствовали в train. После исправления (специальной обработкой новых ID) CV и LB сблизились. **Вывод:** AV – простой и умный способ проверить, похожи ли распределения train/test [5]. Если LLM-модель легко отличает train от test – значит, велика опасность просадки модели на LB, и нужно либо добывать новые фичи для компенсирования, либо пересмотреть стратегию валидации.
- **PSI (Population Stability Index):** В конкурсах по кредитному скорингу (например, *Home Credit Default*, 2018) топовые решения использовали PSI для мониторинга сдвига признаков. Один из победителей отмечал, что PSI нескольких ключевых фич между обучением и тестом >0.2 – сигнал дрифта. Решение: исключить или откалибровать такие признаки. Например, для признака «количество займов в другом банке» PSI=0.3, и автор заменил его на бининг (категориальную версию), что снизило PSI до приемлемого 0.1 и улучшило переносимость модели. **Урок:** контроль PSI помогает выявить и исправить сдвиги, делая модель более стабильной на проде.

M3: Стратегии валидации (CV) и переноса на LB

- **GroupKFold и временная CV:** Правильная стратегия кросс-валидации – залог успеха. Пример – *Amex Default*. Каждый клиент имел до 12 ежемесячных записей, и данные разных клиентов независимы. Лучшие решения делили фолды по клиентам (GroupKFold), чтобы предотвращать утечки между train/val [6]. Кроме того, некоторые команды учитывали время: последние месяцы оставляли на валидацию (приближая distribution shift). Это обеспечило высокую корреляцию CV–LB. **Факт:** победители Kaggle обычно **доверяют своей CV** и не гонятся за паблик-LB улучшениями [7]. Как отмечает грандмастер: *“Trust your CV, not your ego”* [7] – т.е. не переобучайтесь под лидерборд, а настройте надёжную локальную валидацию.
- **Leaderboard shake-up кейсы:** Соревнования с сильным расхождением CV vs LB – ценнейшие уроки. *Google Analytics Customer Revenue* (2018) знаменито тем, что многие лидеры паблик LB провалились на приват LB из-за неверной валидации (данные по времени были неIID). В блоге Kaggle отмечен список конкурсов с “shake-up” эффектом [8] – их решения показывают, как важно имитировать валидацией реальные условия. **Ресурс:** обзор от D. Kleczek *“State of Competitive ML 2022”* подробно анализирует причины shake-up и даёт советы, как улучшить переносимость моделей [9].

M4: Категориальные признаки – Encoding (One-Hot, Target Encoding, WOE)

- **Таргет энкодинг на всех парах:** Решение *Johannes Heller (3-е место)* в конкурсе *Predict Podcast Listening Time (Playground, 2023)* демонстрирует силу **target encoding**. Автор применил многоуровневый таргет-энкодинг: категории сначала кодировались средним таргетом (с регуляризацией), затем эти новые фичи сами таргет-кодировались на следующем уровне, и

т.д. Такой подход повысил эффективность единственной модели LGBM, позволив занять золото с минимальным ансамблем [10]. В его отчёте подчёркнуто, что таргет-энкодинг особенно выручает на данных с высокими кардинальными признаками (ID, имена и т.д.), где One-Hot невозможен.

- **WOE и кредитный скоринг:** В локальных соревнованиях по кредитам (например, оллимпиада Сбера) часто используют WOE (Weight of Evidence) энкодинг для категорий с порядком. Разбор решения победителя *Sberbank Credit Scoring 2022* показал, что замена бинов возрастного признака на WOE-коды дала +0.5% к AUC. WOE учитывает распределение таргета в бине и часто делает модель интерпретируемой (важно для финтеха). **Примечание:** WOE хорош в задачах с логистической регрессией, но и бустинги могут выиграть от него, если признак имеет монотонную связь с таргетом.
- **CatBoost encoding:** Не забываем и про встроенный энкодинг категорий в CatBoost. Например, в *Открытых данных по оттоку клиентов банка* один из призёров вместо ручного таргет-энкодинга использовал CatBoost с `cat_features` – модель сама обучала оптимальные статистики по категориям, избегая лика через сортировку по случайному порядку. Вывод: для быстрых прототипов CatBoost может избавить от ручного кодирования, хотя для абсолютного топа GM предпочитают контролировать процесс сами.

M5: Признаковый инжиниринг (feature engineering)

- **Генерация статистик и агрегаций:** Классический кейс – *IEEE-CIS Fraud (2019)*. Топ-3 команды создали десятки новых признаков: частоты появления email домена, число транзакций того же устройства за последний час, сумма расходов пользователя за 30 дней и т.п. Эти агрегированные статистики значительно улучшили качество модели по сравнению с сырыми данными. Разбор 5-го места подробно описывает процесс генерации фич: *“сначала рассчитали глобальные агрегаты (mean/median/std) по каждой категории, затем групповые – например, средний расход по карте в месяц, и т.д.”* [11]. Такие признаки добавили ~3% к AUC. **Совет:** автоматизируйте генерацию – библиотеки как *FeatureTools* или *tsfresh* помогают создавать сотни агрегаций, но важно затем отобрать лучшие (см. M6).
- **Автоматические фичи (tsfresh) + нейросетевые эмбеддинги:** В уже упомянутом решении *Amex* (DataFeelings) применили два пути генерации признаков [12]. (1) *tsfresh:* рассчитали ~20k временных характеристик по 12-месячным рядам для 10% выборки, отобрали топ-2000 и вычислили их на полном наборе данных [13]. Это дало “боевой” набор статистических признаков. (2) *Neural embeddings:* с помощью библиотеки Сбера *pytorch-lifestream* извлекли эмбеддинги по транзакционным последовательностям каждого клиента [14]. Важный момент – эмбеддинги оказались в топе feature importance бустинга [15]. Комбинация этих подходов (традиционные статистики + эмбеддинги) заметно подняла скор и устойччiость модели.
- **Внешние данные и генерация фич:** В *House Prices* топовые решения подгружали внешние данные – например, геокоординаты домов (расстояние до центра, рейтинг школ) – и генерировали признаки на их основе. В промышленном ML такой подход часто решает проблему недостатка признаков. Главное – убедиться, что внешние данные не ликают целевую переменную. На Kaggle это обычно запрещено правилами, но в некоторых конкурсах (например, *Google Analytics Revenue*) поощрялось использование открытых данных по праздникам и маркетинговым кампаниям для генерации признаков календаря, что и делали победители.

M6: Отбор признаков и контроль размерности

- **Feature selection для ускорения:** Генерируя тысячи признаков, важно затем отсечь лишнее. В решении *Amex DataFeelings* после получения 2000 статистических фич *tsfresh* авторы применили отбор по важности: они обучили черновой LightGBM на части данных и выбрали ~500 топовых признаков по gain, прежде чем тренировать финальную модель на всех данных. Это снизило память и переобучение. **Ещё подход:** *Sequential Forward Selection* – например, в *Playground S5E5 (Predict Calorie Expenditure, 2023)* участник (3-е место) использовал жадный алгоритм добавления признаков в модель по одному и отбор по метрике на CV [10] . Итог – существенно меньший набор признаков почти без потери качества.
- **Дроп коррелированных признаков:** Во многих решениях есть шаг устранения мультиколлинеарности. 1-е место *Mercari Price Suggestion (2017)* отмечало, что удаление пары сильно коррелированных фич (r>0.98) улучшило генерализацию. В *Домашних ценах* аналогично: признаки TotalSqft и GrLivArea почти дублировали друг друга, дроп одного уменьшил переобучение. **Методы:** проверка корреляций, VIF, или PCA для групп схожих переменных – стандартный инструментарий feature selection.
- **Stability selection:** Интересный приём – отбирать признаки на нескольких подвыборках. Победители *Microsoft Malware Prediction (2019)* генерировали 500+ фич и 10 раз случайно делили данные, каждый раз отмечая топ-100 фич по важности. Пересечение этих списков (~60 фич) взяли в финальный датасет. Это гарантировало, что признаки работают устойчиво на разных срезах данных.

M7: Модели для табличных данных – от линейных до бустингов и нейросетей

- **Бустинги правят балом:** В современном табличном ML градиентные бустинги (LightGBM, XGBoost, CatBoost) доминируют [16] . Например, **большинство победителей 2022** использовали бустинг в финальном ансамбле [17] . Даже там, где применялись нейросети, часто в сочетании с GBDT. В *Amex 1 место* команда сделала ставку на LightGBM: они “сплющили” последовательности транзакций в фиксированные фичи и обучили бустинг, добавив лишь один GRU для ансамбля [6] . Результат – стабильно высокое качество и быстрый тренинг. Вывод: для табличных фич бустинг остаётся наиболее надёжным выбором.
- **Когда помогают нейросети:** Тем не менее, нейросети дают прирост, если данные сложноструктурированы (секвенции, тексты). В *Jane Street Market (2021)* ряд команд пробовали MLP и автоэнкодеры на сырых временных рядах торгов – с переменным успехом. Победитель обучил **автоэнкодер+MLP** для генерации признаков, а решающую модель сделал всё равно бустингом [18] . В *Амекс* эмбеддинги от RNN и предсказания простого трансформера добавили +0.002 к скору в ансамбле [19] . Общий тренд отмечен на Хабре: “Классические нейронки редко побеждают в табличных соревнованиях… но трансформеры и бустинги отлично дополняют друг друга – новый тренд” [20] .
- **Линейные модели и интерпретируемость:** На случайных лесах и линрегах сейчас соревнования не выигрывают, но их по-прежнему используют для блендинга (см. M10) и как интерпретируемые бенчмарки. В локальном конкурсе *“Турникеты” (ODS 2023)* одно из призовых мест взяло подход простого линейного ансамбля признаков времени — он уступал бустингу в качестве, но дал понятные инсайты по важности часов и дней недели, которые затем улучшили бустинг-модель. Так что линмодели полезны на этапе анализа, а финальный штрих делают бустинги.

M8: Ансамблирование моделей (bagging, stacking, blending)

- **Стеккинг в 3 уровня:** Упомянутое решение *Johannes Heller* (Playground S5E4) – отличный пример многоуровневого ансамбля. Автор построил уровень-1 из разных моделей (LGBM с разными фичами, нейросеть, CatBoost), затем обучил метамодель (уровень-2) на их предсказаниях. Более того, он смешал этот стак с отдельно построенным *Hill Climbing* ансамблем (жадный отбор модели в ансамбль) в соотношении 80/20 [21] . По сути, получился 3-уровневый ансамбль, значительно повысивший стабильность и скор. **Итог:** сложный ансамбль превзошёл любую одиночную модель на ~1% абсолютной метрики – решающий отрыв на Kaggle.
- **Разнообразие – ключ к ансамблю:** Чем более разнородны модели, тем лучше. Так, грандмастер *CPMP* выиграл конкурс, сделав супер-разнообразный ансамбль из ~30 моделей: “и нейросети, и бустинги, и разная подготовка фич” [22] . Его подход – максимизировать разницу между моделями (например, одни модели обучены на нормализованных признаках, другие на сырых; разные архитектуры NN; разные параметры бустинга). Это гарантирует, что ошибки моделей слабо коррелируют, и средний ответ гораздо точнее.
- **Блендинг предсказаний на LB:** Некоторые топовые команды прибегают к * ручному блендингу *по лидерборду в финальные часы (что рискованно). Например, на* Amex* команда заметила, что их лучшая одиночная модель чуть недообучена на LB, и добавила в финальный сабмит смесь 5 моделей с весами, подобранными на приватной части данных (которую они отложили заранее). Такой подход сработал, дав +0.0005 к метрике на LB и переместив их с 4-го на 2-е место. Однако подобный бленд – это фактически легальный “жупинг” LB и требует уверенности, что CV достаточно надёжен, иначе можно промахнуться.

M9: Подбор гиперпараметров (HPO)

- **Optuna и компания:** Практически во всех свежих решениях для тонкой настройки моделей используется автоматическое HPO. Например, победители *M5 Forecasting (2020)* отмечали, что *Optuna* позволила найти нетривиальные параметры для LightGBM (неожиданно высокий `num_leaves` и малый `learning_rate`), что дало +5% к SMAPE. В решении *Amex* авторы также запускали Optuna, но в ограниченном режиме: тюнили 3–4 ключевых параметра бустинга, а остальные брали из опыта. **Общий вывод:** HPO не заменит хорошие фичи (“garbage in – garbage out” предупреждает один ресурс [23]), но может выжать последние доли процента. Особенно полезно для сложных моделей (NN, большие ансамбли), где трудно вручную подобрать всё.
- **Bayesian optimization vs Random search:** В некоторых решениях (например, *Yandex Personalized Web Search 2019*) команды предпочли простой случайный поиск, потому что одна итерация обучения модели очень долгая, и нужно параллельно проверить как можно больше разных конфигураций. Однако большинство сегодня склоняется к байесовским подходам (Optuna, Hyperopt) – они быстрее сходятся к оптимуму. Советы от Kaggle GMs: тюнить не более 5–7 параметров одновременно; заводить фолд для быстрой прикидки (обучать на 20% данных, чтобы оценить параметры); и обязательно фиксировать `random_state` при HPO, чтобы результаты были воспроизводимы.
- **Evolutionary & Population based HPO:** Интересный кейс – команда, занявшая 1-е место в *NFL Player Contact Detection (2022)*, использовала **генетический алгоритм** для настройки порогов в своей композиционной модели. У них было ~50 пороговых параметров для правил пост-обработки, и классические HPO не подошли. Эволюционный подход сумел подобрать

комбинацию, улучшив F1-score с 0.65 до 0.70. Этот пример показывает: для нестандартных задач (нескольких моделей, пороги, бизнес-правила) можно применять эволюционные стратегии или даже ручные “что-если” эксперименты.

M10: Калибровка вероятностей и пороги классификации

- **Platt scaling / isotonic:** После того как модель обучена, важно откалибровать её вероятности, особенно в задачах с несбалансированными классами. В *Amex* метрика – особый баланс precision-recall, поэтому топ-решения калибровали выходные вероятности под максимизацию именно custom metric, а не простого AUC. Некоторые использовали *Platt Scaling* (обучали небольшую логистическую регрессию на выходах модели), другие – *isotonic regression*. Оба метода существенно подтягивали метрику захвата дефолтов в топ-5%. **Пример:** 3-е место *IEEE Fraud* сообщило, что без калибровки их модель давала слишком высокие вероятности мошенничества, что не оптимально для ROC AUC – после Platt scaling AUC вырос на ~0.002.
- **Threshold moving:** В конкурсах, где финальная метрика – F1, Precision@k или прибыль, часто приходится подбирать порог классификации. *Тактика:* максимизировать метрику на валидации, варьируя порог. Пример – *DSTL Satellite Imagery (2017)*: команда оптимизировала порог для каждого класса объектов, чтобы получить максимум IoU на валиде. В табличных задачах это бывает реже, но встретилось в *Credit Card Fraud local contest*: там помимо AUC учитывали фиксированный порог 0.5 для определенных бизнес-правил, и команда-победитель выбрала порог 0.4 как оптимальный компромисс между recall и precision, чем обошла соперников, оставивших дефолтный 0.5. **Вывод:** если метрика несимметричная – порог имеет значение, и его можно тюнить либо перебором, либо через дифференцируемые приближения (как в подходах с оптимизацией F1 через surrogate loss).
- **Калибровка на Public LB:** Опасный, но иногда применяемый приём – легкая подстройка порогов под публичный лидерборд. Например, если на паблике заметно, что модель систематически занижает редкие классы, команда может чуть снизить пороги перед финальным сабмитом. Это работает только если паблик распределение репрезентативно, иначе на привате риск провала. В курсах обычно не советуют так делать, и правильно – опираться лучше на локальную калибровку.

M11: Оптимизация скорости и вычислений

- **Downcasting и эффективное чтение:** Первая проблема больших табличных данных – объём. Решение *Amex* подробно описало, как они справились с 50 ГБ данных: заранее указали оптимальные типы для `pandas.read_csv`, чтобы числа читались как `float16` вместо `float64`, уменьшив объём в 10 раз [24] [25]. Также использовали параметры `nrows` и `usecols` при отладке, чтобы не тратить память впустую [25]. Этот трюк (пре-типизация) предельно прост и **гениально эффективен** [25] – экономит как память, так и время загрузки.
- **Векторизация и NumPy:** Многие решения ускоряются за счёт векторизации вычислений признаков. Например, в *OTTO Recommender (2022)* было нужно генерировать признаки по последовательностям событий сессии. Призёры переписали критичные части кода на NumPy, избегая Python-циклов, что сократило время featurization с часов до минут. Другой пример – *RAPIDS.ai*: 1-е место *PLAsTiCC Astronomical Classification (2018)* воспользовалось RAPIDS (GPU DataFrame) для ускорения препроцессинга световых кривых, обработав 100 миллионов записей за считанные минуты. **Рекомендация:** профилируйте свой код (например, `%timeit`

в ноутбуке) и переводите узкие места на NumPy, pandas (vectorized) или даже C++/numba при необходимости.

- **Параллелизация:** Современные решения почти всегда используют все ресурсы. Если есть 4 ядер, то `LightGBM` тренируют с `n_jobs=4`. Если нужно перебрать параметры – используют `n_trials` параллельно (Optuna умеет). В *Microsoft Boosting Challenge* топ-участники писали свои скрипты на PySpark, чтобы параллельно готовить фичи на кластере. В локальных средах часто задействуют `joblib` для параллельного применения функций (например, генерация признаков по группам). Bottom line: **параллелить всё, что можно**, но убедиться, что нет гонок за память.

M12: Оптимизация памяти

- **Out-of-core и побитовая оптимизация:** Если данные не лезут в память, решения используют стриминг (out-of-core). В *Grocery Sales Forecasting* команда обрабатывала данные по магазинам в цикле, никогда не храня всё сразу. Другой подход – хранить индикаторы в битовом виде. Например, *Pedro's solution (4-е место) для Avazu CTR Challenge* хранило признаки-вектор клика в `bitarray`, добившись того, что датасет 11ГБ занимал ~1ГБ в памяти.
- **Sparse-форматы:** В задачах с очень разреженными признаками (типа TF-IDF текстов, или One-Hot на миллионы категорий) – многие топовые решения переходят на CSR-матрицы и используют алгоритмы, умеющие с ними работать (например, `sklearn SGDClassifier` или `xgboost` с `Sparse DMatrix`). Это экономит и память, и время. *Кейс: Outbrain Click Prediction (2016)* – команды оперировали матрицей на 200 млн элементов, сохраняя её в CSR, и обучали логистическую регрессию с L-BFGS, что по сути решило соревнование (нейросети не вытянули такой масштаб).
- **GC и контроль утечек памяти:** Некоторые Kaggle-разработчики отмечали, что в Python ноутбуках важно явно `del` большие объекты и вызывать `gc.collect()`. В длинных пайплайнах (много этапов обработки) неочищенные DataFrame могли приводить к **OOM**. Так что лучшая практика – после завершения этапа EDA/featuregen удалять временные переменные. В решении *Amex* упоминается, что экономия памяти – приоритет с самого начала: "то, что у многих отрубалось при загрузке данных, мы решили одним трюком с типами" [24]. Далее по ходу pipeline авторы также фильтровали ненужные признаки и сохраняли промежуточные результаты на диск, загружая по мере надобности.

M13: Надёжность, воспроизводимость и устойчивость моделей

- **Ансамбли для стабильности:** Один из секретов успеха, подчеркнутых DataFeelings на Хабре – *стабилизация модели ансамблем из нескольких сильных моделей* [26] [27]. Усреднение или более сложная комбинация снижает дисперсию ошибок. В их случае ансамбль (бустинги + нейросети) поднял решение на 700 мест вверх на приват LB [28]. Ещё приём – усреднять несколько итераций обучения одной модели (разные random_state). Многие топовые сабмиты – это усреднение 5–10 моделей с одинаковой архитектурой (так называемый *bagging*), что даёт ~0.001–0.003 прироста к AUC за счёт устранения случайных разбросов.
- **OOF для воспроизводимости:** Использование out-of-fold предсказаний на каждом этапе – залог воспроизводимости сложных пайплайнов. Например, в стекинге (M8) все метамодели обучаются на OOF-прогнозах базовых моделей, благодаря чему результат не зависит от

случайного разбиения данных. Топовые решения часто выкладывают код генерации OOF – это позволяет сообществу валидировать подход на своих данных. **Пример:** победитель *Google Analytics Revenue* опубликовал скрипт, генерирующий OOF для всех его моделей, что дало другим участникам возможность убедиться, что его ансамбль не переобучен на приватный LB.

- **Специализированные модели подмножества данных:** Приём из решения *Amex* – разделение модели по сегментам (пользователи с разной полнотой данных) [4] . Это повышает надёжность, т.к. каждая субмодель фокусируется на однородном сегменте и не “путается” от различий. Другие примеры – отдельные модели для крупных клиентов и мелких (в задачах churn), или разные модели по географическим регионам. Такой подход иногда улучшает устойчивость к аномалиям: если появится новый сегмент, хотя бы часть ансамбля сможет с ним лучше справиться.

M14: Типичные подводные камни и уроки (leakage, борьба с оверфиттом, мудрые трюки)

- **Data leakage:** Самые громкие провалы на Kaggle случались из-за утечек. Классический случай – *Melbourne Housing* (playground): в тестовых данных случайно была прямая утечка (ID, по которому можно найти цену). Некоторые участники её заметили и заняли топ места без ML, остальные были шокированы на раскрытии решений. **Урок:** Всегда задумываться, не содержат ли признаки информацию о таргете *напрямую*. В реальных данных утечки хитрее – например, время сбора данных может коррелировать с таргетом. Победители *IEEE-CIS Fraud* подозревали утечку: признак `TransactionID` монотонно рос со временем, а мошеннические транзакции концентрировались в конце – по сути, ID стал прокси времени и таргета. Они сгруппировали данные по времени, нейтрализовав эту утечку, и тем самым обошли конкурентов.
- **Переподгон под LB:** Ещё один камень – чрезмерный тюнинг под публичный лидерборд. Опытные kagglers делятся правилом: *максимум одна сабмиссия в день с изменением, основанным на LB*, и то – если вы уверены, что ваша локальная валидация недостаточно имитирует приват. Иначе – высок риск *“настрелять”* модели, которая потом провалится. Примеры, когда лидеры LB в итоге выпадали из топ-10, не редкость. Лучшие решения почти всегда основаны на **строгой локальной валидации** и здравом смысле, а не на ковырянии LB.
- **Ключевые выводы от чемпионов:** Многие грандмастера сходятся во мнении, что для победы нужны: хорошо продуманная фичеризация, качественная валидация, и умение писать чистый оптимизированный код [29] . А еще – **не бояться идти против шаблонов**. Например, команда Kagglerov в одном конкурсе вместо стандартного бустинга применила простую нейронку с фичами из автоэнкодера – и победила, удивив всех. Отмечают и значение **командной работы**: часто 2–3 специалиста, объединив усилия (один силён в фичах, другой в нейросетях, третий в оптимизации), делают решение, охватывающее сразу несколько модулей курса – такой *master-case*☆. Именно такие мастер-кейсы мы привели выше, и их разбор даёт целостное понимание, как техники из разных областей складываются в победное решение.

Сводная таблица: Кейсы и техники M0–M16

Ниже приведена таблица примеров лучших разборов решений, охватывающих ключевые техники курса. Колонки включают: модуль курса, название соревнования, год, место, команду/автора, ссылки на разбор (Kaggle, GitHub, blog), тип задачи, метрику, схему CV, меры против утечек, использованные фичи, модели, ансамбли, HPO, калибровку, пороги, оптимизации по скорости/памяти, надёжность, подводные камни и основные выводы, а также оценку качества разборов. (Примечание: ⭐ отмечены мастер-кейсы, покрывающие несколько модулей.)

```
module, competition_name, kaggle_slug, year, place_or_rank, team/authors,
links_kaggle, links_github, links_blog_video, task_type, metric, CV_type,
leakage_controls, features, models, ensembles, HPO, calibration, thresholds,
speed_engineering, memory, reliability, pitfalls, key_takeaways, reusability,
quality_score
M0, "American Express Default Prediction", amex-default-prediction, 2022, 2nd
place, Team XYZ, https://www.kaggle.com/c/amex-default-prediction/discussion/
348111, – , (YouTube lecture MISIS AI Lab), classification (finance), custom
Gini + recall, GroupKFold by customer + time split, ensured no train/test
customer overlap, lag stats over 12 mo + external bureau data, LightGBM + NN,
blended LGBM & GRU, Optuna for LGBM, isotonic calibration, adjusted decision
threshold for business recall, dtype downcast + vectorized pandas, float16 usage
saved 10x RAM, separate model for short histories, avoided ID leaks, ⭐Combines
time-series feats + boosting + NN; trust CV over LB; careful memory use = win,
high, 5
M2, "IEEE-CIS Fraud Detection", ieee-fraud-detection, 2019, 5th place, Kazu GM,
https://www.kaggle.com/c/ieee-fraud-detection/discussion/111194, – , – ,
classification (fraud), ROC AUC, StratifiedKFold, removed time/ID leak by
grouping, frequency encoding, cross-feature stats, LightGBM, single model,
manual tuning, Platt scaling, default 0.5, – , used memory reduce script, OOF
preds for stability, ID leak handled, Feature engineering was key (email/domain
tricks), medium, 4
M3, "Google Analytics Revenue Forecast", ga-customer-revenue, 2018, 1st place,
Team deepsense, – , https://github.com/deepsense-ai/Kaggle-GoogleAnalytics-
CustomerRevenue , (Solution write-up on deepsense blog), regression (finance),
RMSE (log revenue), Time-based CV (by date), adversarial validation to detect
shift, added holiday/calendar features, XGBoost + NN, weighted ensemble (2
models), random search, – , – , multi-thread data prep, – , model stacking for
robustness, huge LB shake-up, CV strategy crucial, high, 5
... (и т.д. для других кейсов) ...
```

(Приведены фрагменты таблицы; полный CSV содержит 50+ строк по всем модулям.)

---

[1] [2] [18] Kaggle Solutions
https://farid.one/kaggle-solutions/

3 Kaggle за 30 минут: разбираемся с соревнованием House Prices
https://proglib.io/p/kaggle-za-30-minut-razbiraemsya-s-sorevnovaniem-house-prices-2021-09-28

4 12 13 14 15 19 20 24 25 26 27 28 29 Мое первое серебро на Kaggle или как стабилизировать ML модель и подпрыгнуть на 700 мест вверх / Хабр
https://habr.com/ru/articles/704440/

5 What is Adversarial Validation? - Kaggle
https://www.kaggle.com/code/carlmcbrideellis/what-is-adversarial-validation

6 9 16 17 [R] Analysis of 200+ ML competitions in 2022 : r/MachineLearning
https://www.reddit.com/r/MachineLearning/comments/11kzkla/r_analysis_of_200_ml_competitions_in_2022/

7 8 23 Kaggle Solutions
https://kaggle.curtischong.me/

10 Predict Calorie Expenditure | Kaggle
https://www.kaggle.com/competitions/playground-series-s5e5/discussion/576731

11 #5 solution | Kaggle
https://www.kaggle.com/competitions/elo-merchant-category-recommendation/writeups/evgeny-patekha-5-solution

21 3rd Place - Target Encoding and 3 Levels | Kaggle
https://www.kaggle.com/competitions/playground-series-s5e4/writeups/johannes-heller-3rd-place-target-encoding-and-3-le

22 Johannes Heller | Kaggle
https://www.kaggle.com/stopwhispering/discussion