NFL Big Data Bowl 2026:
Предсказание движения игроков

Проект по курсу Machine Learning
Задача: Предсказать координаты (x, y) игроков NFL во время полёта мяча после паса
Метрика: RMSE по координатам x и y

$$RMSE = \sqrt{\frac{1}{2N}\sum_{i=1}^{N}\left((x_{true,i} - x_{pred,i})^2 + (y_{true,i} - y_{pred,i})^2\right)}$$

Постановка задачи

Что дано:
Tracking data игроков ДО броска мяча (позиция, скорость, ускорение, ориентация)
Место приземления мяча (ball_land_x, ball_land_y)
Информация о целевом ресивере (player_to_predict)

Что предсказываем:
Координаты x, y каждого игрока в каждом кадре ПОСЛЕ броска
10 кадров в секунду, среднее время полёта ~1.1 секунды

Бизнес-ценность:
Анализ тактических схем защиты
Оценка качества пасов и маршрутов
Предсказание вероятности перехвата

Описание датасета

Параметр	Значение
Input данные	~4.9M строк
Output данные	~0.56M строк
Уникальных игр	272
Уникальных розыгрышей	~14,000
Уникальных игроков	2,157
Недель данных	18

Объём данных:

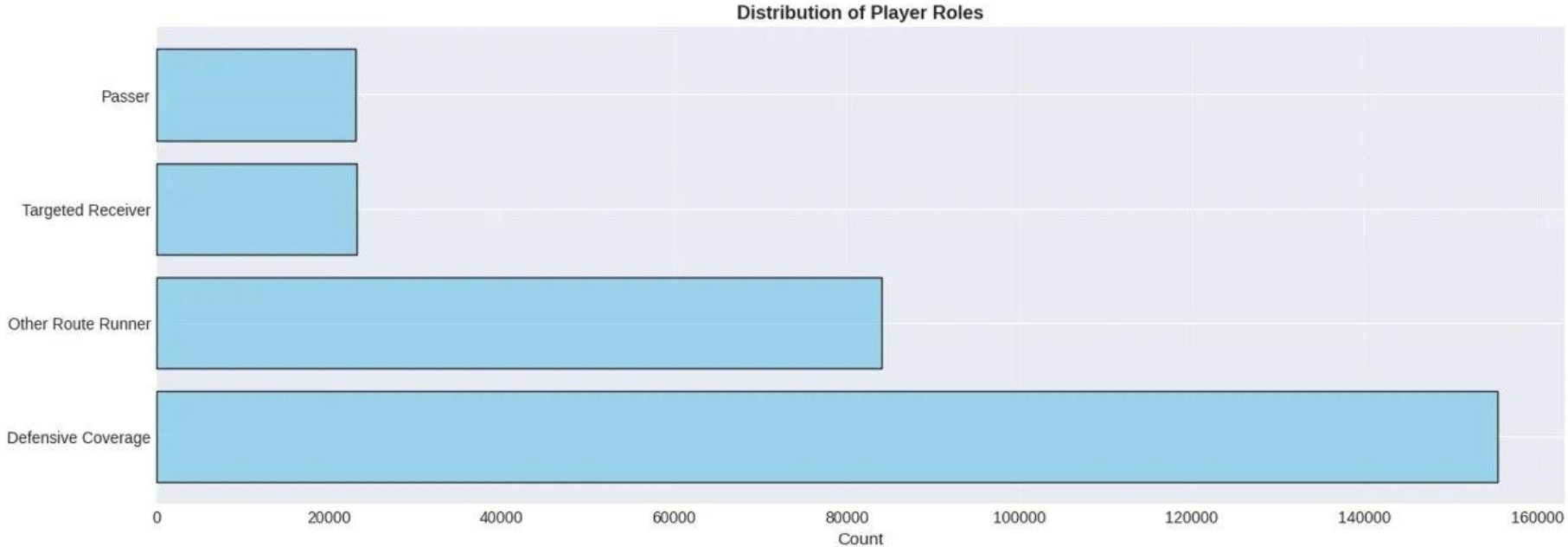
Временная компонента: ✅ frame_id (10 FPS)
Пространственные признаки: ✅ координаты x, y на поле
Датасет полностью соответствует требованиям проекта

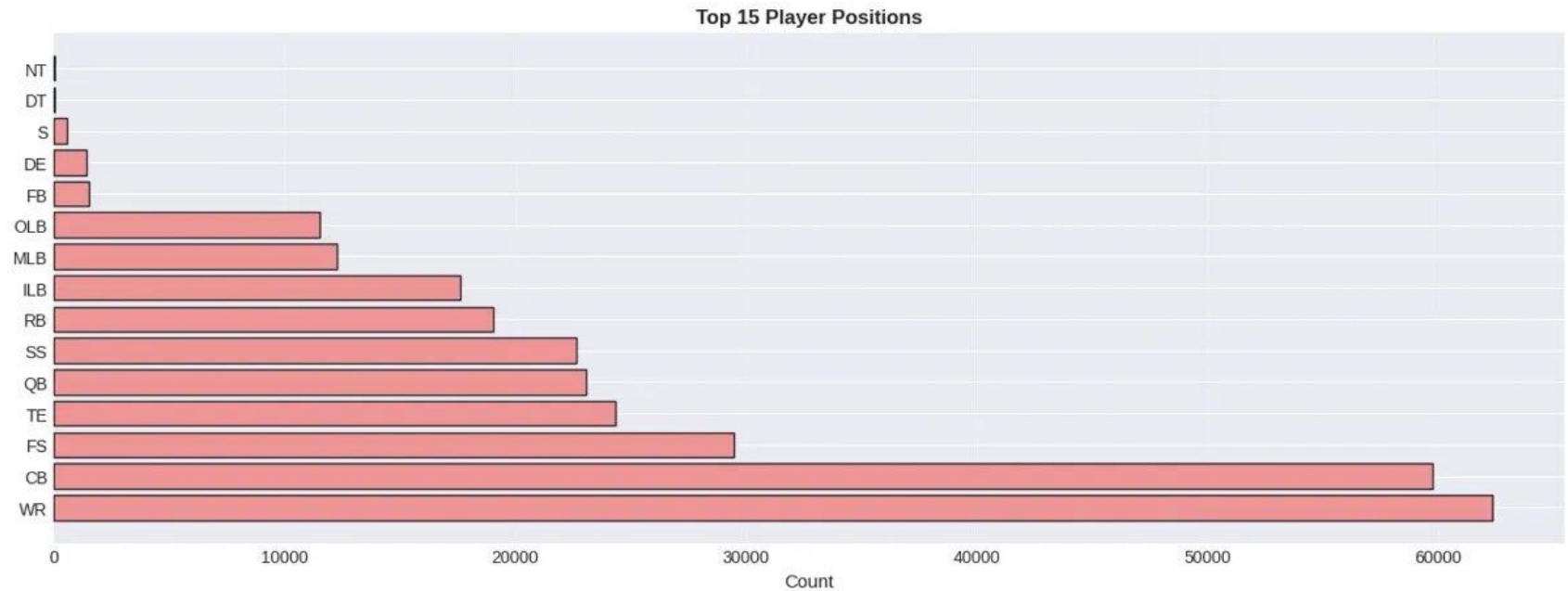
EDA — Распределение ролей игроков

Ключевые наблюдения:

Defensive Coverage: 155,397 записей (доминирует) — защитники активно реагируют на пас
Other Route Runner: 84,063 — другие принимающие, создающие отвлечение
Targeted Receiver: 23,151 — целевой принимающий (главный объект предсказания)
Passer: 23,103 — квотербек, бросающий мяч

Инсайт: Модель должна по-разному обрабатывать разные роли — целевой ресивер движется к мячу, защитники реагируют на него


Distribution of Player Roles
Passer
Targeted Receiver
Other Route Runner
Defensive Coverage
0
20000
40000
60000
80000
100000
120000
140000
160000
Count

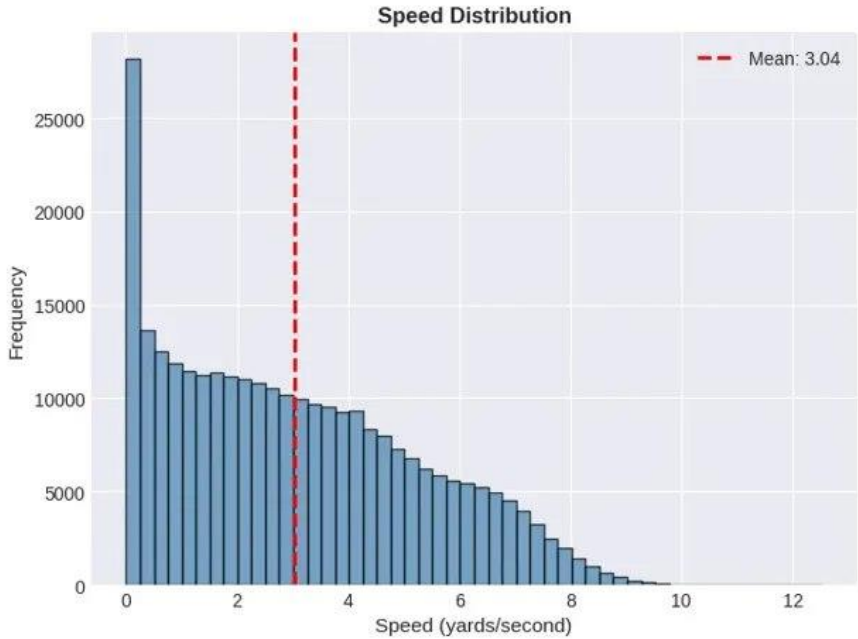

Top 15 Player Positions
NT
DT
S
DE
FB
OLB
MLB
ILB
RB
SS
QB
TE
FS
CB
WR
0
10000
20000
30000
40000
50000
60000
Count

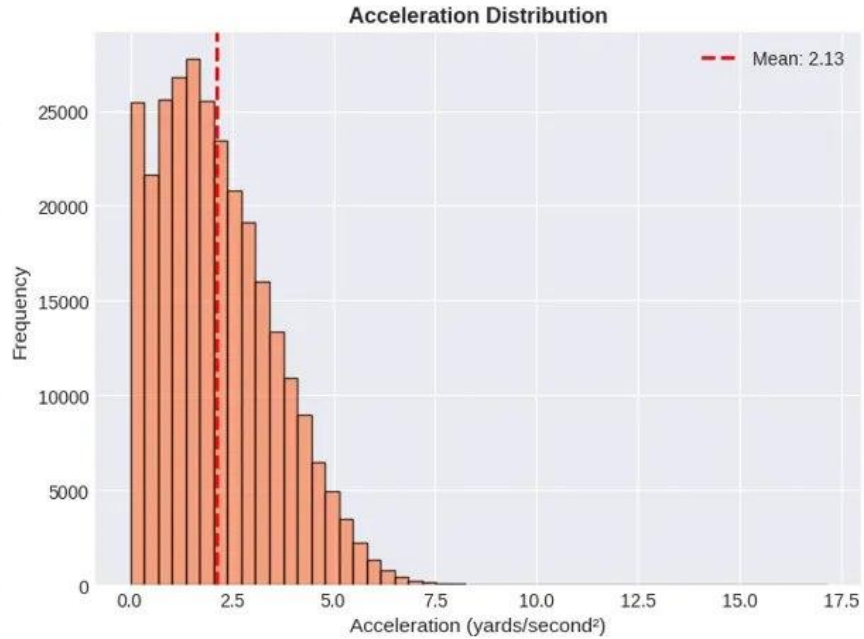
EDA — Скорость и ускорение

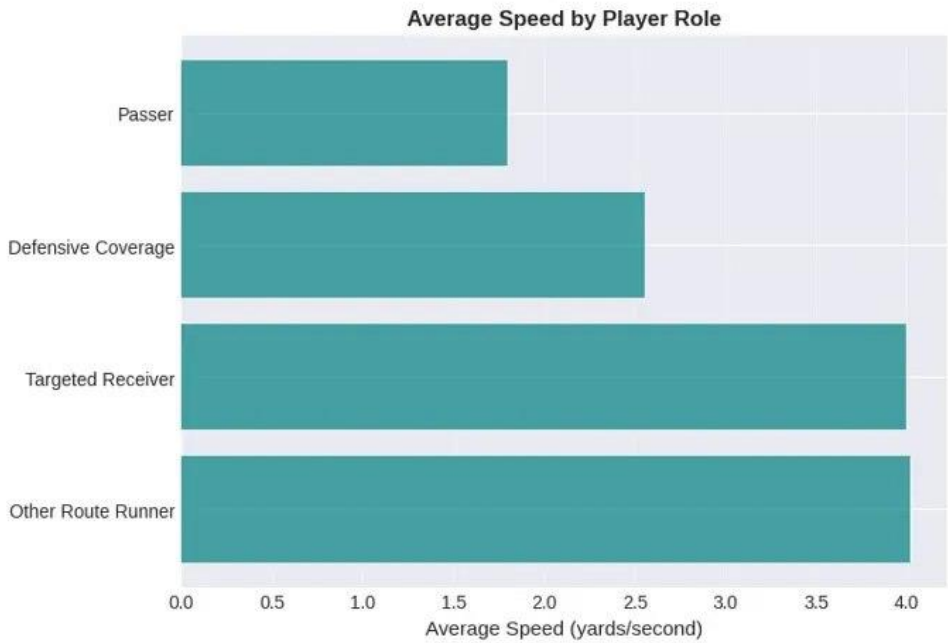
Статистика скоростей:
Средняя скорость: **3.04 ярда/сек**
Распределение: правосторонний скос (много медленных, мало быстрых)
Максимум: ~12 ярдов/сек (спринт)

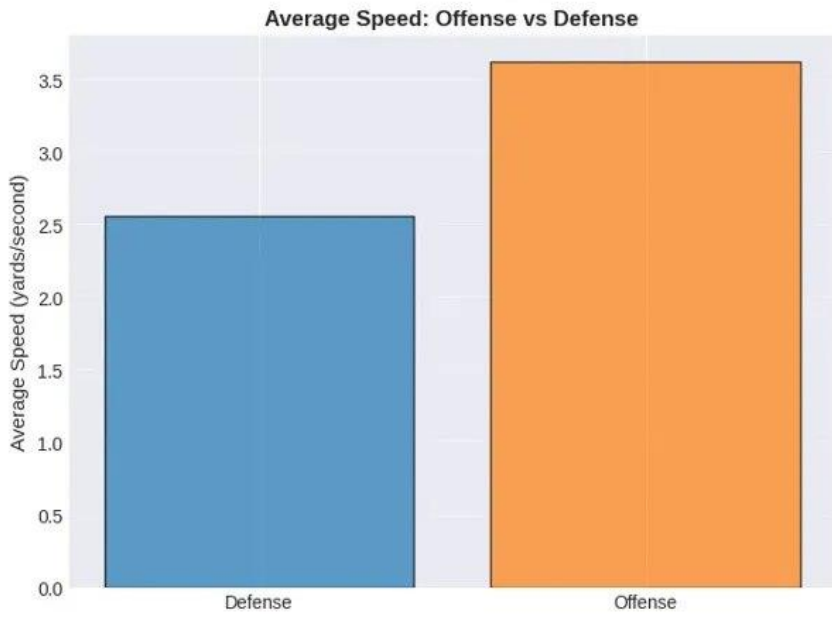
По ролям игроков:
Targeted Receiver и **Other Route Runner**: ~4 ярда/сек (бегут маршруты)
Defensive Coverage: ~3.5 ярда/сек (реагируют на движение)
Passer: ~1.5 ярда/сек (почти неподвижен)

Инсайт: Скорость — ключевой предиктор! Offense быстрее Defense на ~1 ярд/сек


Speed Distribution
Mean: 3.04
Frequency
Speed (yards/second)
0
5000
10000
15000
20000
25000
0
2
4
6
8
10
12


Acceleration Distribution
Mean: 2.13
Frequency
Acceleration (yards/second²)
0
5000
10000
15000
20000
25000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

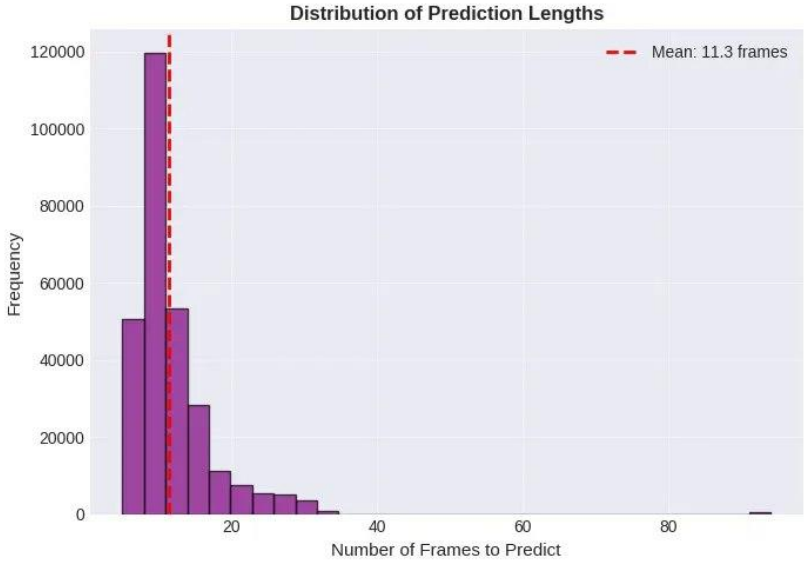

Average Speed by Player Role
Passer
Defensive Coverage
Targeted Receiver
Other Route Runner
Average Speed (yards/second)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

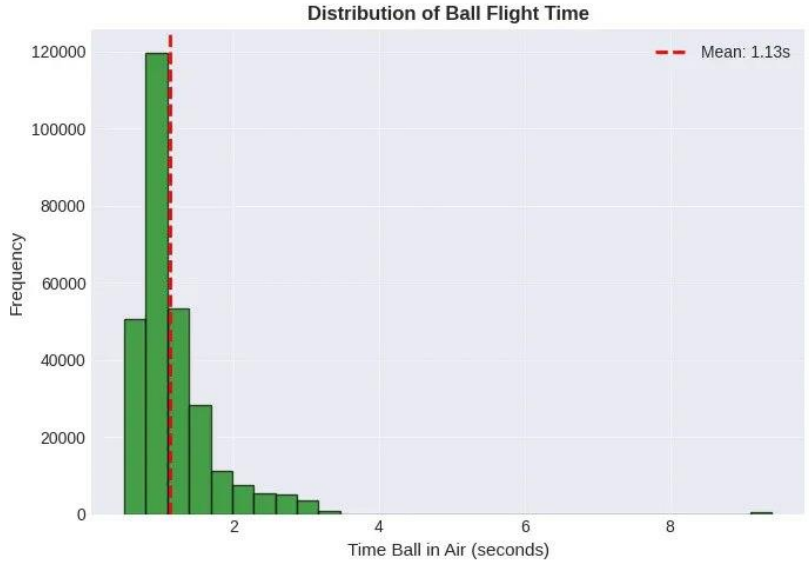

Average Speed: Offense vs Defense
Average Speed (yards/second)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Defense
Offense

EDA — Время полёта мяча

Среднее: **11.3 кадров** (~1.13 секунды)
Медиана: **10 кадров** (1 секунда)
Диапазон: 5–94 кадра (0.5–9.4 секунды)
75% розыгрышей: ≤13 кадров

Инсайт:
Большинство пасов — короткие (quick passes)
Модель должна хорошо работать на коротких последовательностях
Длинные пасы (>30 кадров) — редкие, но сложные случаи4



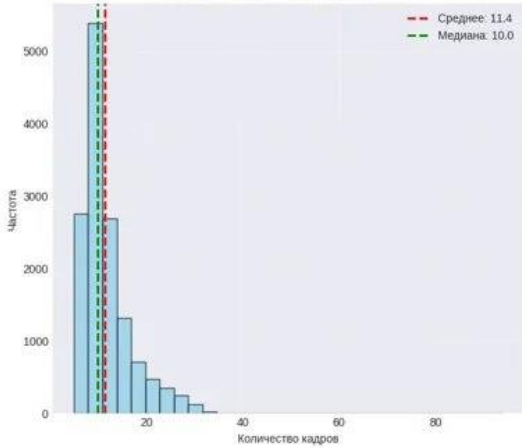


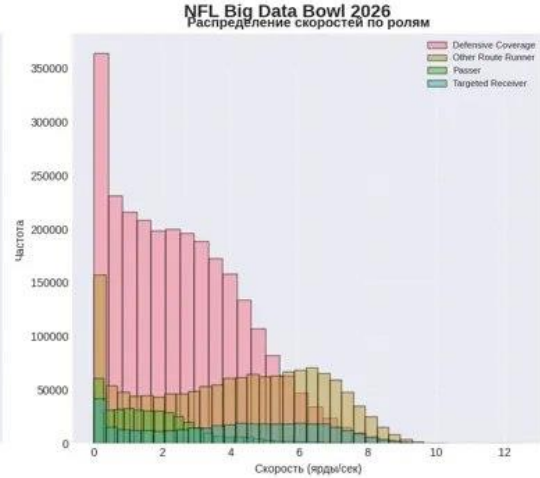
EDA — Позиции на поле

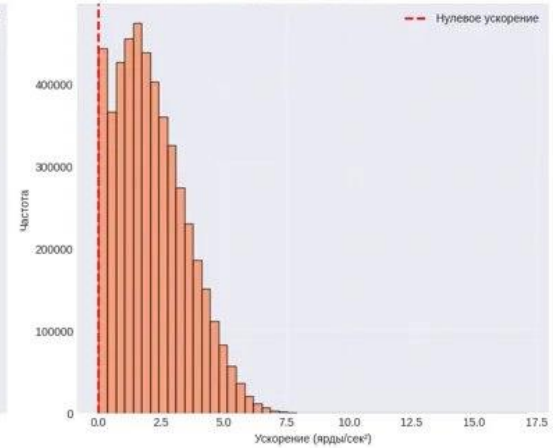
Наблюдения:
Игроки концентрируются в центре поля (y ≈ 26.65 ярдов)
Горизонтальные "полосы" — типичные построения до паса
x координата варьируется широко (зависит от позиции на поле)

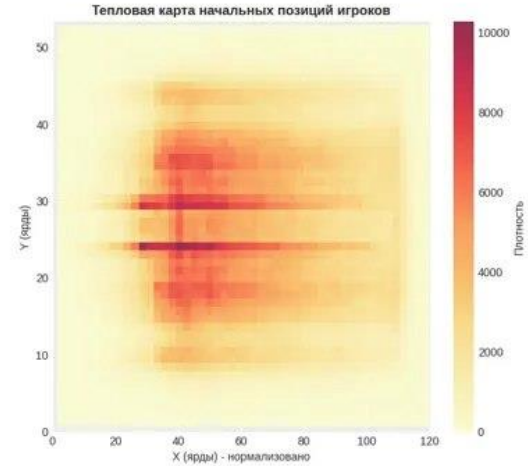
Корреляции (Image 4):
x ↔ ball_land_x: **0.88** (сильная связь!)
x ↔ absolute_yardline: 0**.94**
Скорость и ускорение слабо коррелируют с позицией

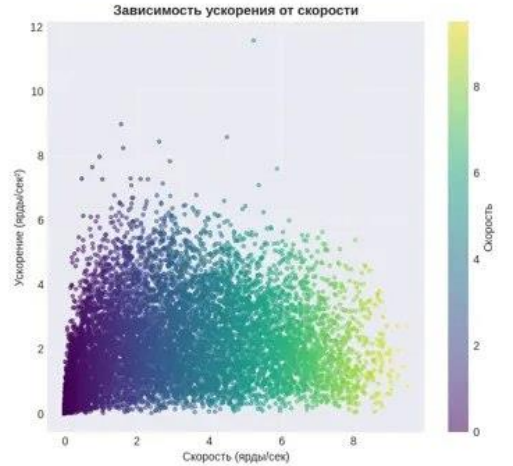
NFL Big Data Bowl 2026

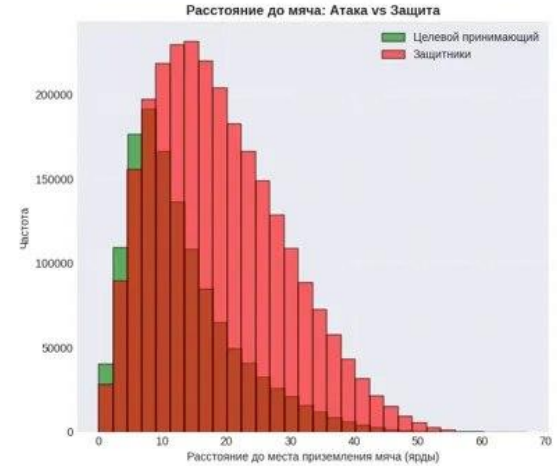


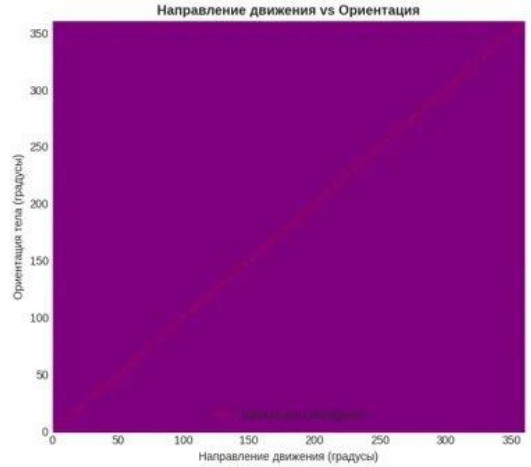


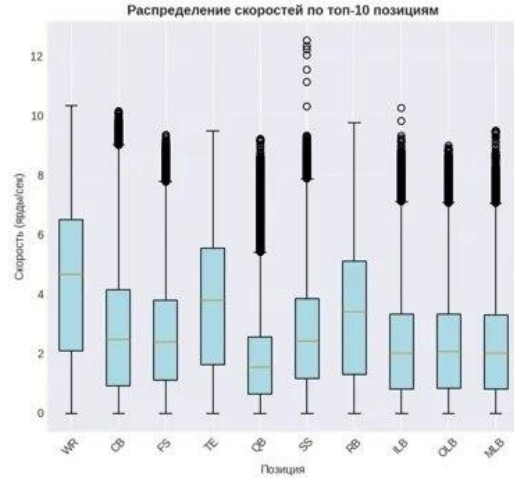


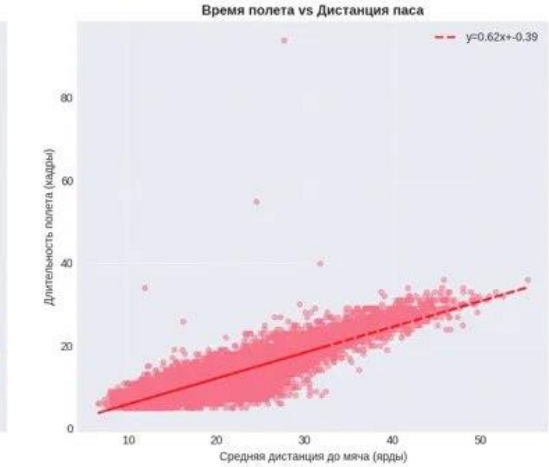








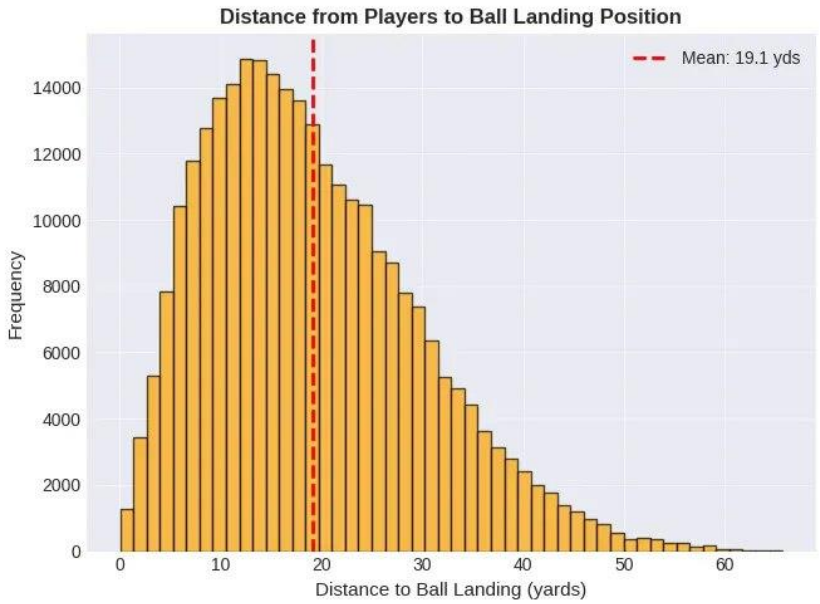


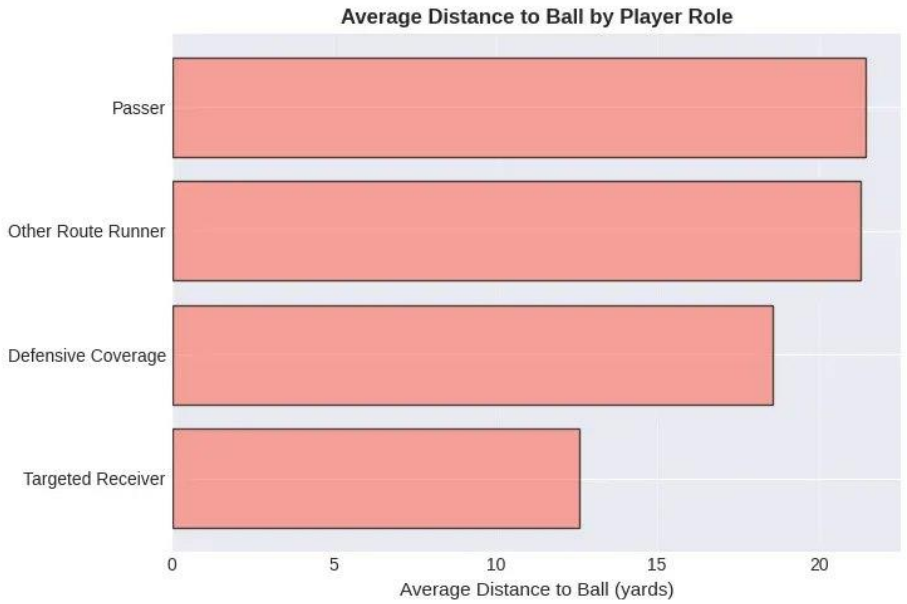


EDA — Расстояние до мяча

Средняя дистанция до места приземления:
Targeted Receiver: ~12 ярдов (ближе всех к мячу)
Defensive Coverage: ~18 ярдов
Other Route Runner: ~20 ярдов
Passer: ~22 ярда (самый далёкий)

Инсайт: Целевой ресивер УЖЕ движется к мячу в момент броска — это ключевая информация для модели!



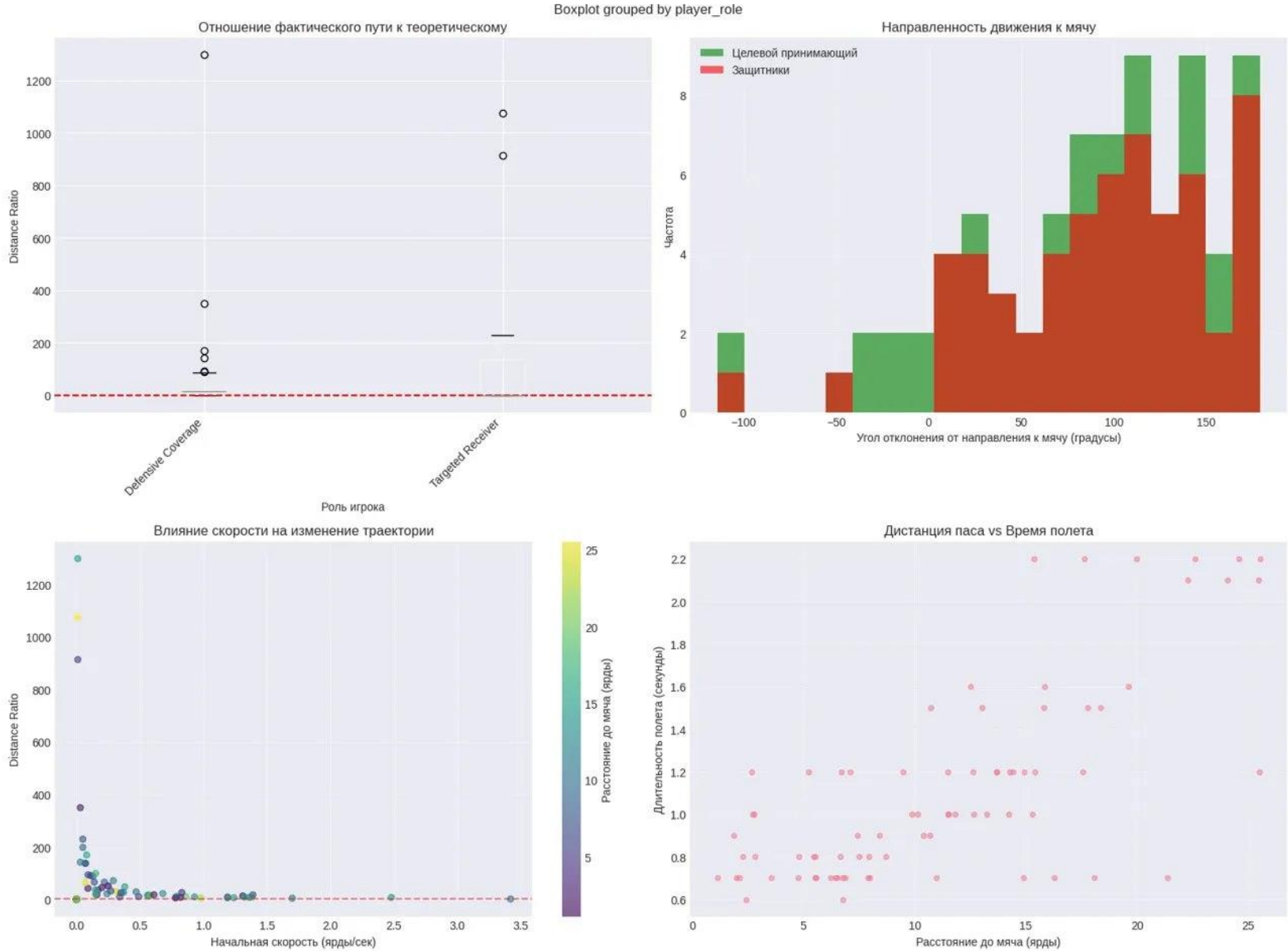


СЛАЙД 9: Анализ физики движения

Проблема выбросов:
Distance Ratio (факт/теория): **77** для ресиверов, **55** для защитников
Ожидаемое значение: 1.0–1.2
Причина: деление на малые скорости (игроки почти неподвижны)

Направленность к мячу:
Целевой принимающий: отклонение **87°** от направления к мячу
Защитники: отклонение 94°

Вывод: Нужна фильтрация выбросов по скорости перед обучением


Boxplot grouped by player_role
Отношение фактического пути к теоретическому
Distance Ratio
Defensive Coverage
Targeted Receiver
Роль игрока
Направленность движения к мячу
Целевой принимающий
Защитники
Частота
Угол отклонения от направления к мячу (градусы)
Влияние скорости на изменение траектории
Distance Ratio
Начальная скорость (ярды/сек)
Расстояние до мяча (ярды)
Дистанция паса vs Время полета
Длительность полета (секунды)
Расстояние до мяча (ярды)

Feature Engineering (Baseline)

Созданные признаки:
Топ признаки по важности:
dist_to_ball (корр. 1.0)
expected_error_to_ball (0.95)
dist_to_ball_y (0.78)

Группа	Признаки
Расстояние	dist_to_ball, vector_to_ball_x/y
Скорость	velocity_x/y, acceleration_x/y
Углы	angle_to_ball, angle_diff_to_ball
Время	estimated_time_to_ball
Позиция	relative_x/y, distance_from_center
Категории	is_targeted_receiver, is_defense, is_offense
Физика	BMI, height_inches, player_weight

Baseline модель

Архитектура:
Две отдельные модели XGBoost: одна для x, одна для y
40 признаков
Train/Val split: 85%/15% по играм

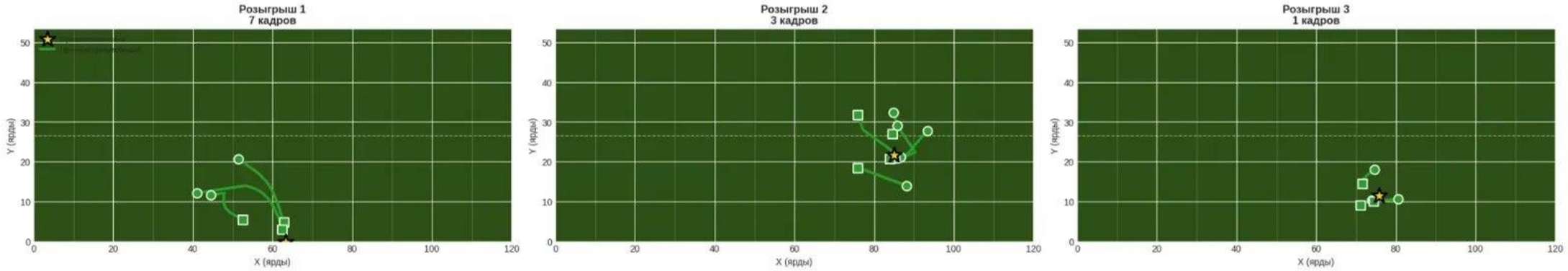
Гиперпараметры:
n_estimators=300, max_depth=8, learning_rate=0.05
subsample=0.8, colsample_bytree=0.8

Метрика	Значение
Validation RMSE	~3.5 ярда
Kaggle RMSE	**3.886 ярда**
Наивный baseline	~5.2 ярда
Улучшение vs наивный	~25%

Визуализация траекторий

Примеры розыгрышей:
Розыгрыш 1 (7 кадров): короткий пас, игроки движутся к мячу
Розыгрыш 2 (3 кадра): очень короткий пас
Розыгрыш 3 (1 кадр): мгновенный пас

Наблюдения:
Зелёные линии — траектории игроков
Жёлтая звезда — место приземления мяча
Квадраты — конечные позиции



Промежуточные метрики

Цель: снизить RMSE до <3.0 ярдов

Этап	Действие	RMSE
Наивный	x + v*t	~5.2
Baseline XGB	**40 признаков**	**3.886**
→ Этап 2	+ аномалии + фичи	?
→ Этап 3	+ SHAP + тюнинг	?

Этап 2 — Цели и задачи

Задачи этапа:
Найти и проанализировать аномалии в данных
Принять решение: удалять или сохранять выбросы
Создать признаки на основе аномалий
Применить ML-методы для поиска сложных выбросов
Расширить feature engineering

Почему это важно для NFL данных:
Экстремальные скорости (рывки, спринты) — реальные игровые ситуации
Необычные позиции на поле могут быть критически важны для предсказания
Баланс: не потерять информацию vs не испортить модель шумом

Статистические методы поиска выбросов

Метод 1: Z-оценка (Z-score)

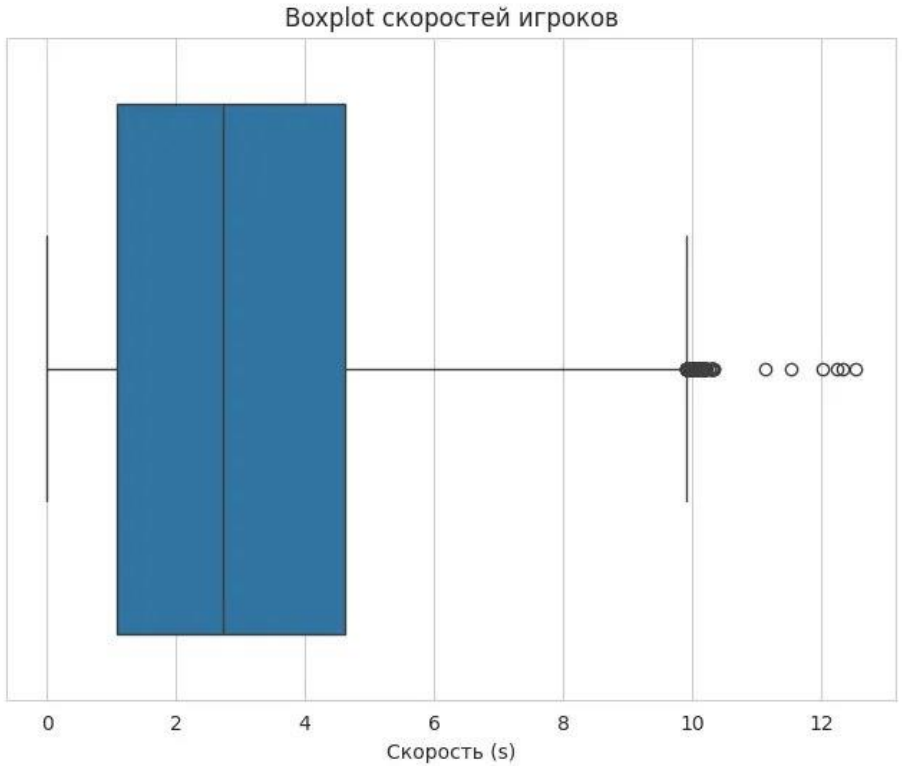
```
pythondf['z_s'] = stats.zscore(df['s'])
outliers_z = df[np.abs(df['z_s']) > 3]
```

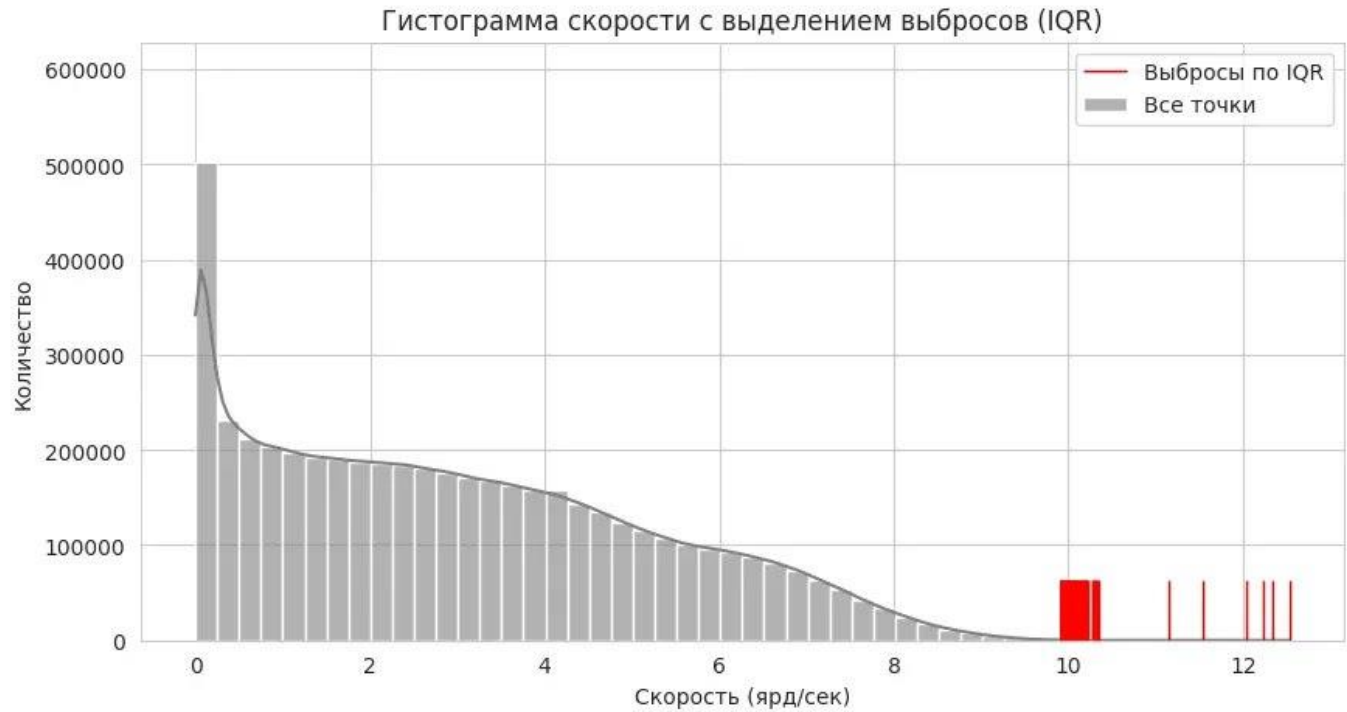
Найдено: **546 выбросов** по скорости
Порог: |Z| > 3 (более 3 стандартных отклонений)

Метод 2: Межквартильный размах (IQR)

```
pythonQ1, Q3 = df['s'].quantile([0.25, 0.75])
IQR = Q3 - Q1
outliers = df[(df['s'] < Q1 - 1.5*IQR) | (df['s'] > Q3 + 1.5*IQR)]
```

Найдено: 1**66 выбросов**
Более консервативный метод


Boxplot скоростей игроков
0
2
4
6
8
10
12
Скорость (s)


Гистограмма скорости с выделением выбросов (IQR)
Выбросы по IQR
Все точки
600000
500000
400000
300000
200000
100000
0
Количество
0
2
4
6
8
10
12
Скорость (ярд/сек)

Анализ выбросов по скорости

Физическая интерпретация:

Скорость > 10 ярдов/сек ≈ **33 км/ч** — это спринт мирового класса
Усэйн Болт: ~12.4 м/с ≈ 13.5 ярдов/сек (абсолютный максимум)
NFL игроки: реалистичный максимум ~10-11 ярдов/сек

Выбросы по признакам:
Вывод: Выбросов мало (<0.1%), но они информативны

Выбросы по признакам:

Признак	Выбросов (Z>3)	% от данных
Скорость (s)	546	0.011%
Ускорение (a)	~1,200	0.025%
x_input	~200	0.004%
y_input	~150	0.003%

Решение по выбросам — НЕ удалять!

Почему мы решили СОХРАНИТЬ выбросы:

Доменная специфика NFL:
Рывки и резкие ускорения — часть игры
Столкновения меняют траектории непредсказуемо
Экстремальные ситуации = критические моменты игры

Информационная ценность:
Модель должна уметь предсказывать и редкие случаи
Удаление может убрать важные паттерны

На**ше решение:**
python# Создаём флаги вместо удаления
df['is_outlier_z_s'] = (np.abs(df['z_s']) > 3).astype(int)
df['is_outlier_iqr_s'] = outliers_iqr.astype(int)
df['is_outlier_s'] = df['is_outlier_z_s'] | df['is_outlier_iqr_s']
Результат: Флаги для 6 признаков (s, a, x, y, o, dir)

ML-метод — Isolation Forest

Настройка алгоритма:

```
pythoniso = IsolationForest(
    n_estimators=200,
    contamination=0.005,  # ожидаем 0.5% аномалий
    random_state=42
)
df_sample["iso_outlier"] = (iso.fit_predict(X) == -1).astype(int)
```

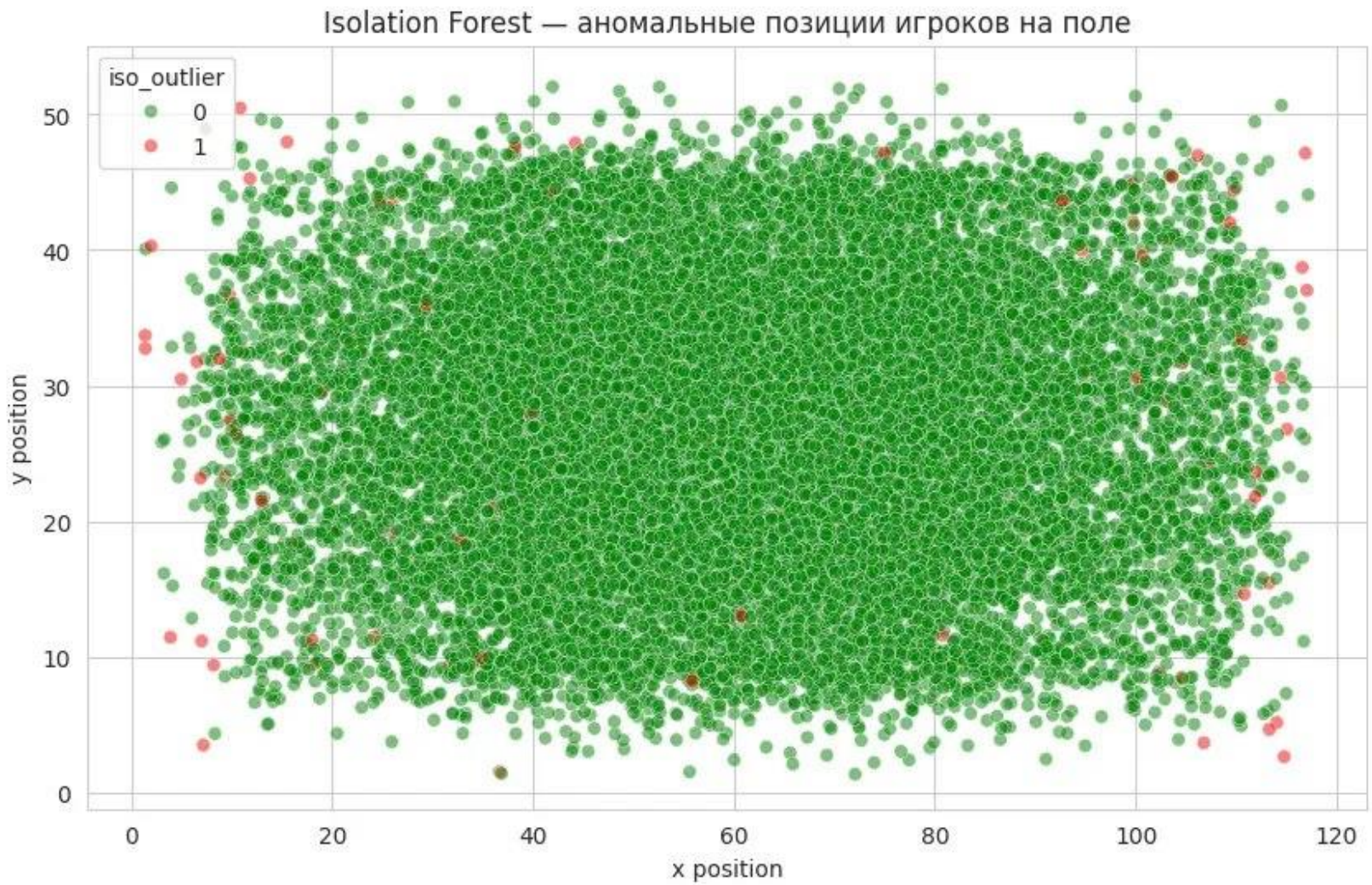
Признаки для детекции:

s, a, x_input, y_input, o, dir

Результаты визуализации:

🟢 Зелёные точки — нормальные наблюдения
🔴 Красные точки — аномалии (края поля)

Инсайт: Isolation Forest выявляет игроков на краях поля — это редкие, но важные ситуации!


Isolation Forest — аномальные позиции игроков на поле
iso_outlier
0
1
y position
x position
0
10
20
30
40
50
0
20
40
60
80
100
120

Интерпретация аномалий Isolation Forest

Что нашёл алгоритм:
Аномалии концентрируются на **краях поля** (y < 5 или y > 48)
Также в **эндзонах** (x < 10 или x > 110)

Почему это важно для модели:
Игрок у боковой линии → ограничен в движении
Игрок в эндзоне → другая тактика защиты
Эти ситуации редки в train, но критичны для предсказания

Решение: Создаём признак iso_outlier как дополнительную фичу

Генерация признаков на основе аномалий

Созданные флаги выбросов:

python# Для каждого признака создаём 3 флага
for col in ['s', 'a', 'x_input', 'y_input', 'o', 'dir']:
    df[f'is_outlier_z_{col}'] = ...   # Z-score > 3
    df[f'is_outlier_iqr_{col}'] = ... # IQR метод
    df[f'is_outlier_{col}'] = ...   # Объединённый флаг

Дополнительные признаки:

Итого: +18 новых признаков на основе аномалий

KNN-признаки (соседи в пространстве)

Признак 1: Плотность окружения (knn_density)

```
pythonknn = NearestNeighbors(n_neighbors=5)
knn.fit(df[['x_input', 'y_input']])
distances, _ = knn.kneighbors(df[['x_input', 'y_input']])
df['knn_density'] = distances.mean(axis=1)
```

Логика: Среднее расстояние до 5 ближайших игроков

Высокая плотность → скученность, сложнее двигаться
Низкая плотность → свобода манёвра

Признак 2: Расстояние до ближайшего защитника

```
pythonknn_def = NearestNeighbors(n_neighbors=1)
knn_def.fit(defense[['x_input', 'y_input']])
distances, _ = knn_def.kneighbors(offense[['x_input', 'y_input']])
offense['closest_defender_dist'] = distances
```

Логика: Чем ближе защитник — тем сложнее принять мяч

Временные признаки

Нормализация фрейма:
pythondf['frame_norm'] = df['frame_id'] /
df['num_frames_output'].clip(lower=1)

Логика: Относительная позиция во времени (0 = начало, 1 = конец)

Циклическое кодирование (sin/cos):
pythondf['frame_sin'] = np.sin(2 * np.pi * df['frame_norm'])
df['frame_cos'] = np.cos(2 * np.pi * df['frame_norm’])

Почему sin/cos:
Линейное кодирование: 0.99 и 0.01 далеко друг от друга
Циклическое: они близки (почти полный цикл)
Модель лучше понимает периодичность движений

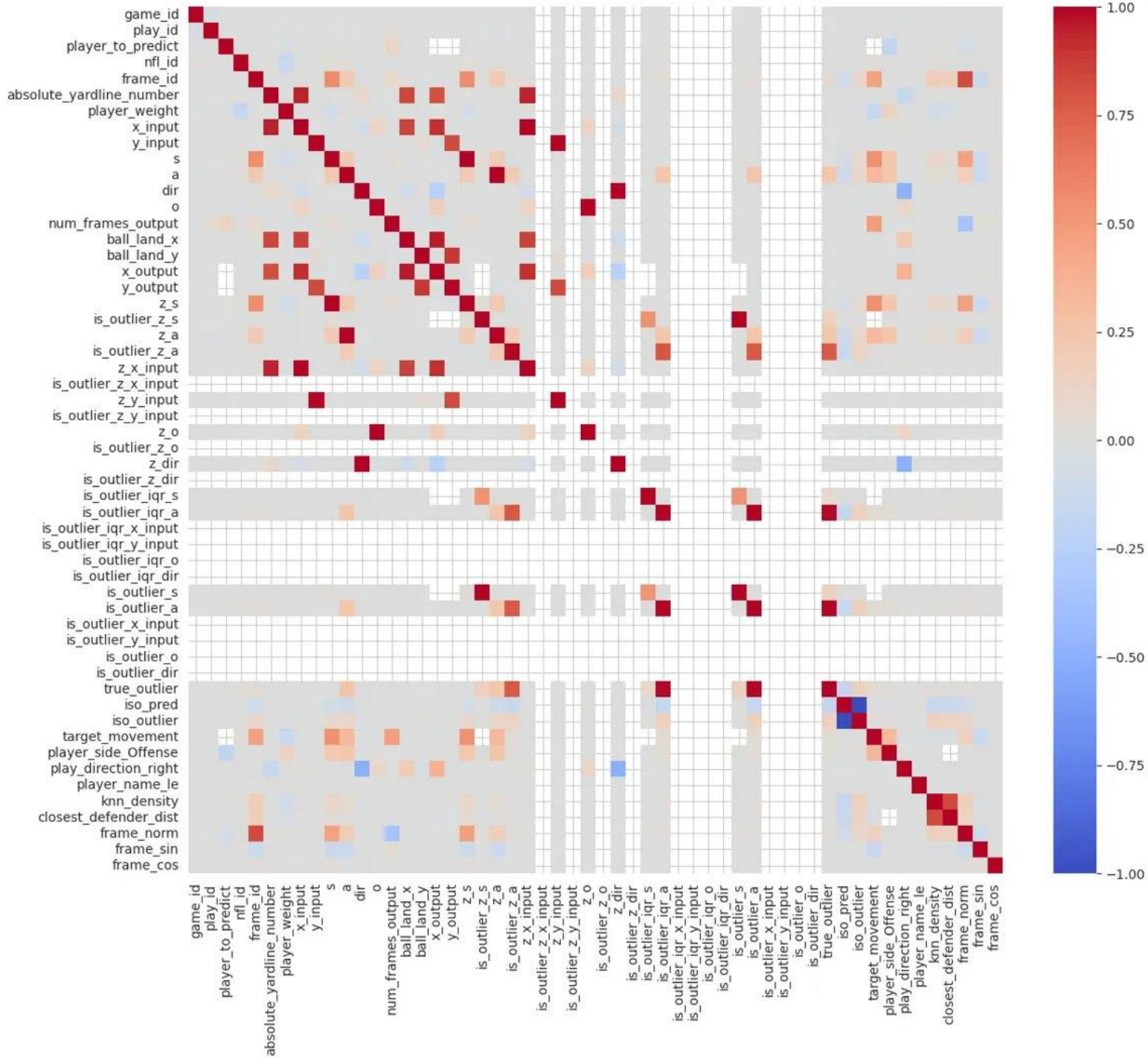
Матрица корреляций расширенного датасета

Сильные корреляции (> 0.7):
x_input ↔ ball_land_x (0.88)
x_input ↔ absolute_yardline (0.94)
x_output ↔ x_input (высокая)

Флаги аномалий:
Слабо коррелируют с основными признаками (хорошо!)
iso_outlier имеет свою уникальную информацию

Но**вые признаки:**
knn_density коррелирует с позицией
frame_sin/cos независимы от координат

Вывод: Новые признаки добавляют информацию, не дублируя старые


game_id
play_id
player_to_predict
nfl_id
frame_id
absolute_yardline_number
player_weight
x_input
y_input
s
a
dir
o
num_frames_output
ball_land_x
ball_land_y
x_output
y_output
z_s
is_outlier_z_s
z_a
is_outlier_z_a
z_x_input
is_outlier_z_x_input
z_y_input
is_outlier_z_y_input
z_o
is_outlier_z_o
z_dir
is_outlier_z_dir
is_outlier_iqr_s
is_outlier_iqr_a
is_outlier_iqr_x_input
is_outlier_iqr_y_input
is_outlier_iqr_o
is_outlier_iqr_dir
is_outlier_s
is_outlier_a
is_outlier_x_input
is_outlier_y_input
is_outlier_o
is_outlier_dir
true_outlier
iso_pred
iso_outlier
target_movement
player_side_Offense
play_direction_right
player_name_le
knn_density
closest_defender_dist
frame_norm
frame_sin
frame_cos
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

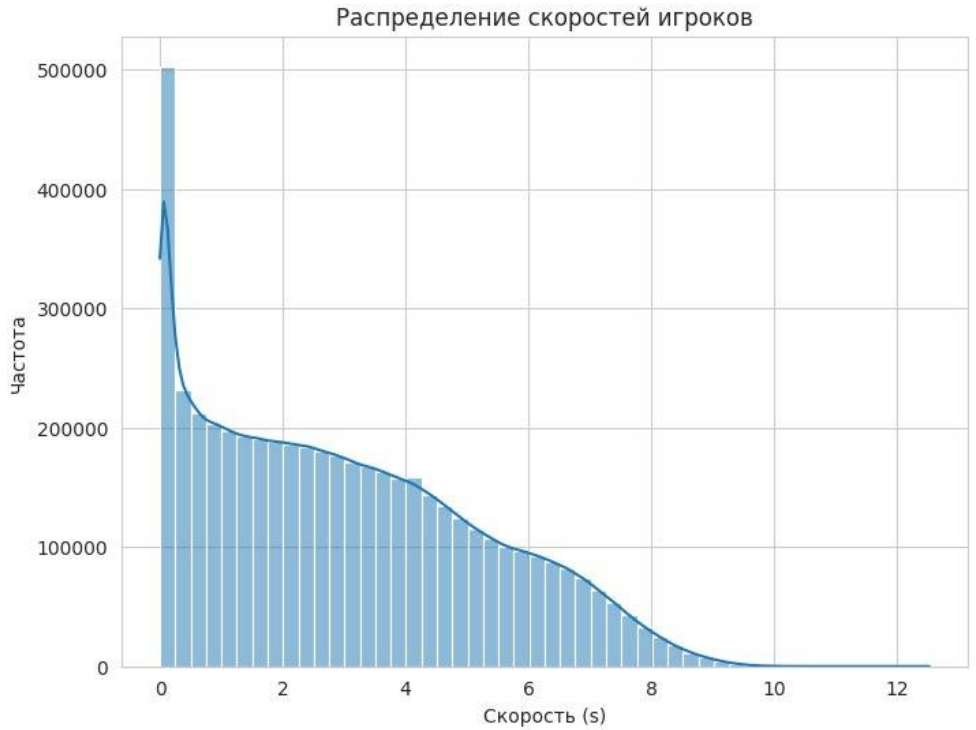
Распределения координат

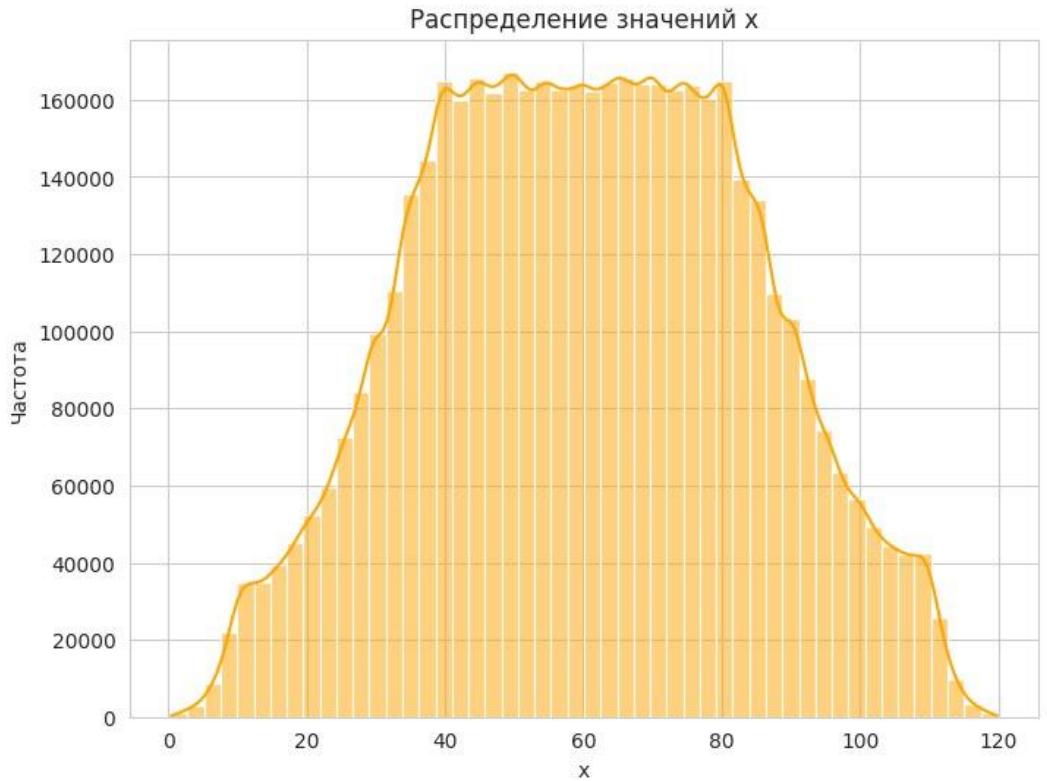
Координата X (длина поля):
Распределение близко к нормальному
Пик в центре поля (40-80 ярдов)
Мало данных в эндзонах

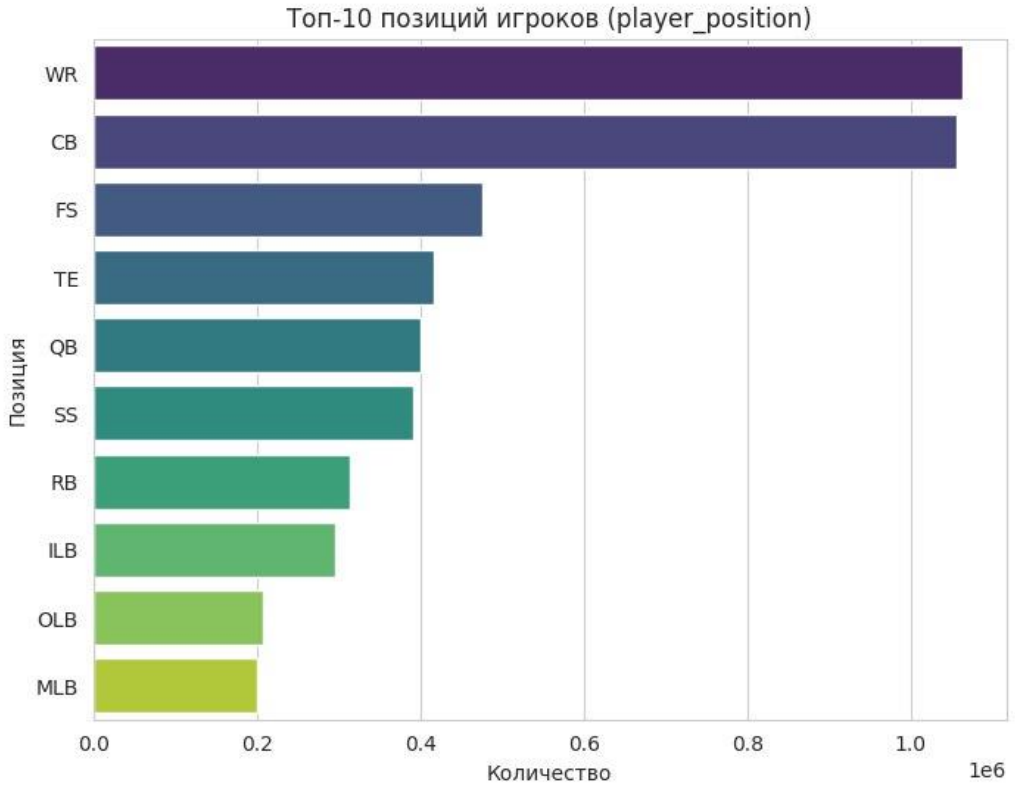
Координата Y (ширина поля):

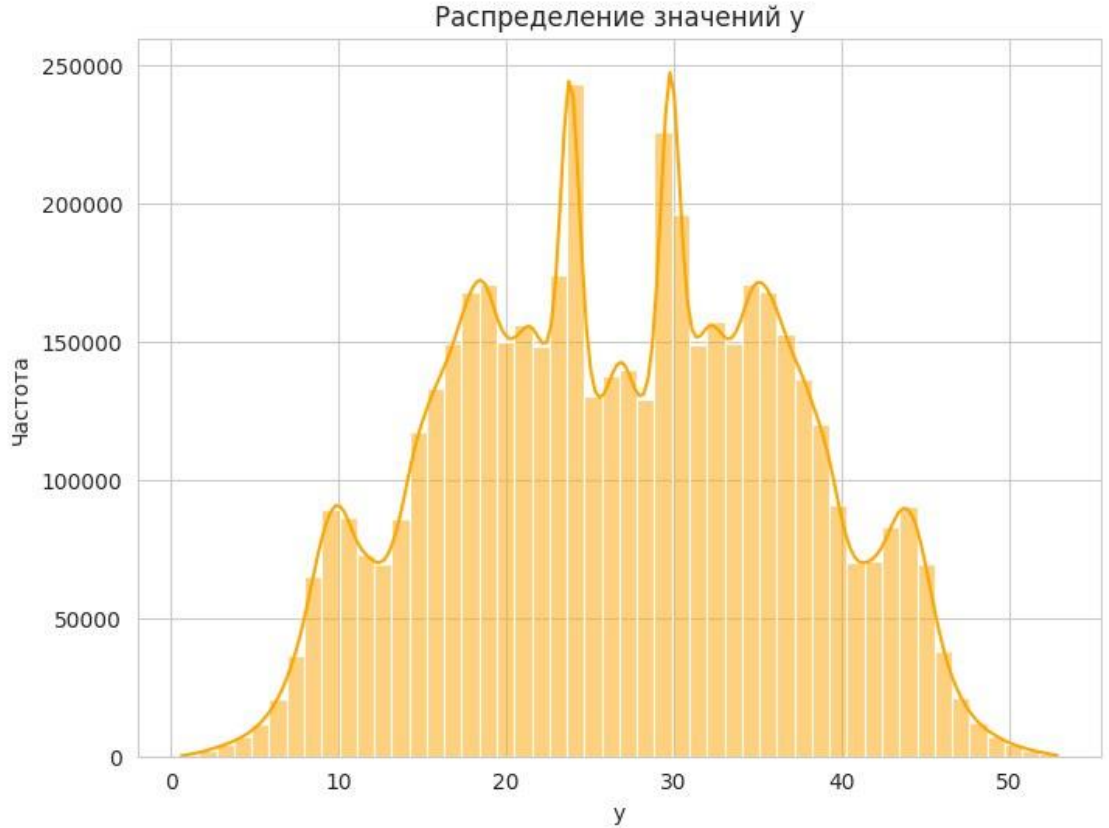
Бимодальное распределение!
Два пика: ~20 и ~35 ярдов
Соответствует типичным построениям (hash marks)

Инсайт: Y-координата имеет сложную структуру — модель должна это учитывать


Распределение скоростей игроков
Частота
Скорость (s)
0
100000
200000
300000
400000
500000
0
2
4
6
8
10
12


Распределение значений x
Частота
x
0
20000
40000
60000
80000
100000
120000
140000
160000
0
20
40
60
80
100
120


Топ-10 позиций игроков (player_position)
WR
CB
FS
TE
QB
SS
RB
ILB
OLB
MLB
Позиция
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Количество


Распределение значений y
250000
200000
150000
100000
50000
0
Частота
0
10
20
30
40
50
y

Группы признаков:

Группа	Кол-во	Примеры
Исходные	10	x, y, s, a, dir, o
Расстояния	5	dist_to_ball, vector_to_ball_x/y
Скорости	4	vx, vy, velocity_x/y
Временные	4	time_remaining, frame_sin/cos
Аномалии	**18**	is_outlier_*, iso_outlier
KNN	**2**	knn_density, closest_defender_dist
Категории	6	is_targeted, is_defense

Выводы по Этапу 2

✅ **Что сделано:**

Поиск аномалий:
Z-score: 546 выбросов по скорости
IQR: 166 выбросов
Isolation Forest: аномалии на краях поля

Решение: Не удалять, а помечать флагами
Сохраняем информацию о редких событиях
Модель сама решит, как использовать

Но**вые признаки:**
18 флагов аномалий
2 KNN-признака (плотность, расстояние до защитника)
3 временных признака (sin/cos кодирование)

📊 **Метрики пока не пересчитывали** — это будет на Этапе 3 с интерпретацией

Переход к Этапу 3

Что дальше:
Обучить модели с новыми признаками
Провести SHAP/LIME интерпретацию
Сравнить важность "старых" и "новых" фичей
Оптимизировать гиперпараметры

Гипотезы для проверки:
Флаги аномалий улучшат предсказание редких случаев
knn_density поможет для скученных ситуаций
closest_defender_dist важен для ресиверов

Этап 3 — Цели и задачи

Задачи этапа:
Глобальная интерпретация моделей (SHAP, LIME)
Локальная интерпретация отдельных предсказаний
Построение SHAP-эмбеддингов и анализ сдвигов
Кластеризация SHAP-пространства
Shapley Flow и граф взаимосвязей
Кросс-валидация с разными наборами признаков
Оптимизация гиперпараметров (Optuna)

Философия этапа:
"Анализ и выводы важнее, чем прирост метрик"

Обученные модели

Оптимальные веса ансамбля:Ridge: 0.03 (мин
имальный вклад)LightGBM: 0.48CatBoost: 0.49

Три класса моделей для сравнения:

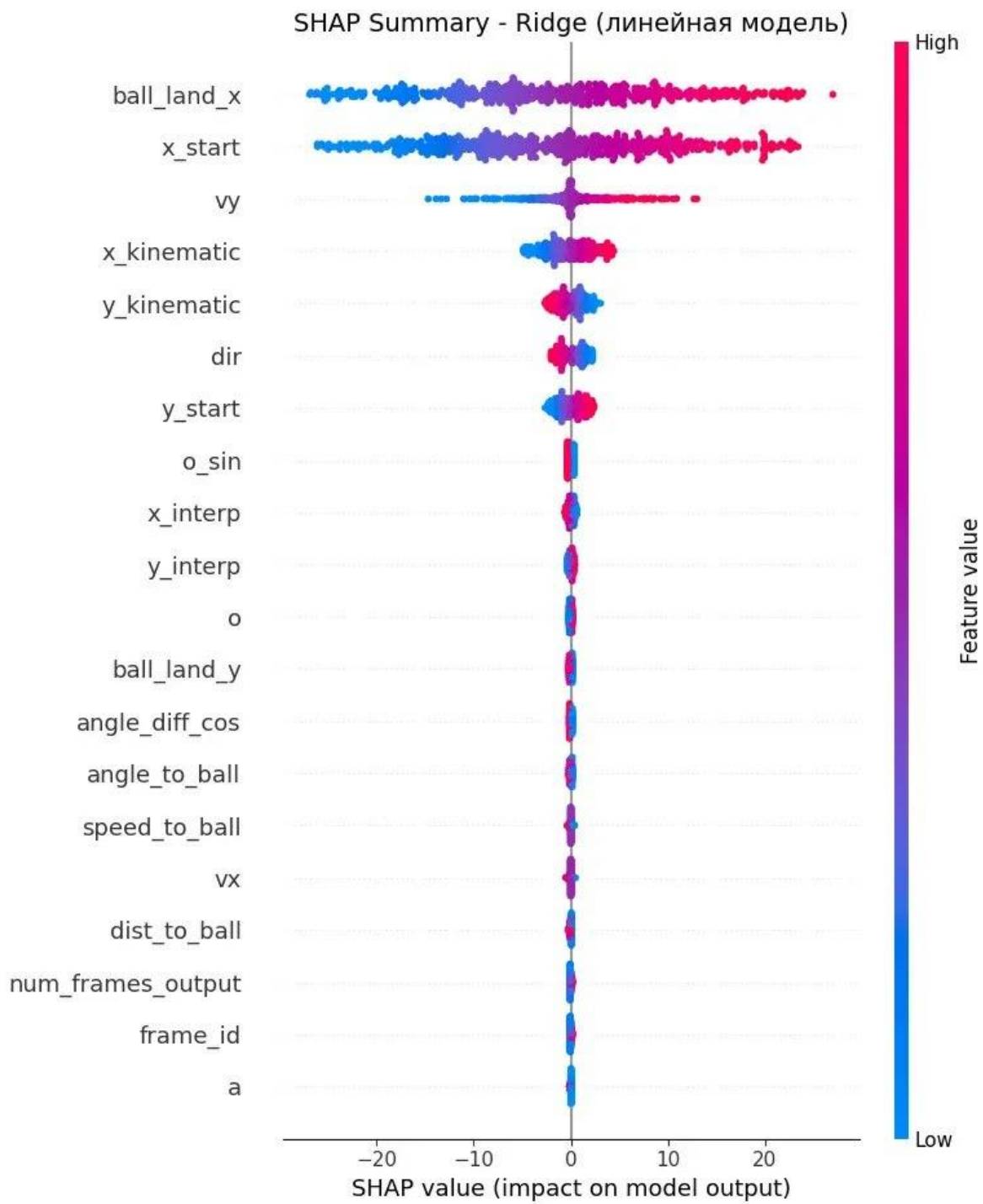
Модель	Тип	RMSE	Особенности
Ridge	Линейная	4.02	Интерпретируемые коэффициенты
LightGBM	Gradient Boosting	3.52	Быстрый, хорошо с табличными данными
CatBoost	Gradient Boosting	3.53	Robust к категориальным
Ensemble	Взвешенный	3.51	Комбинация лучших

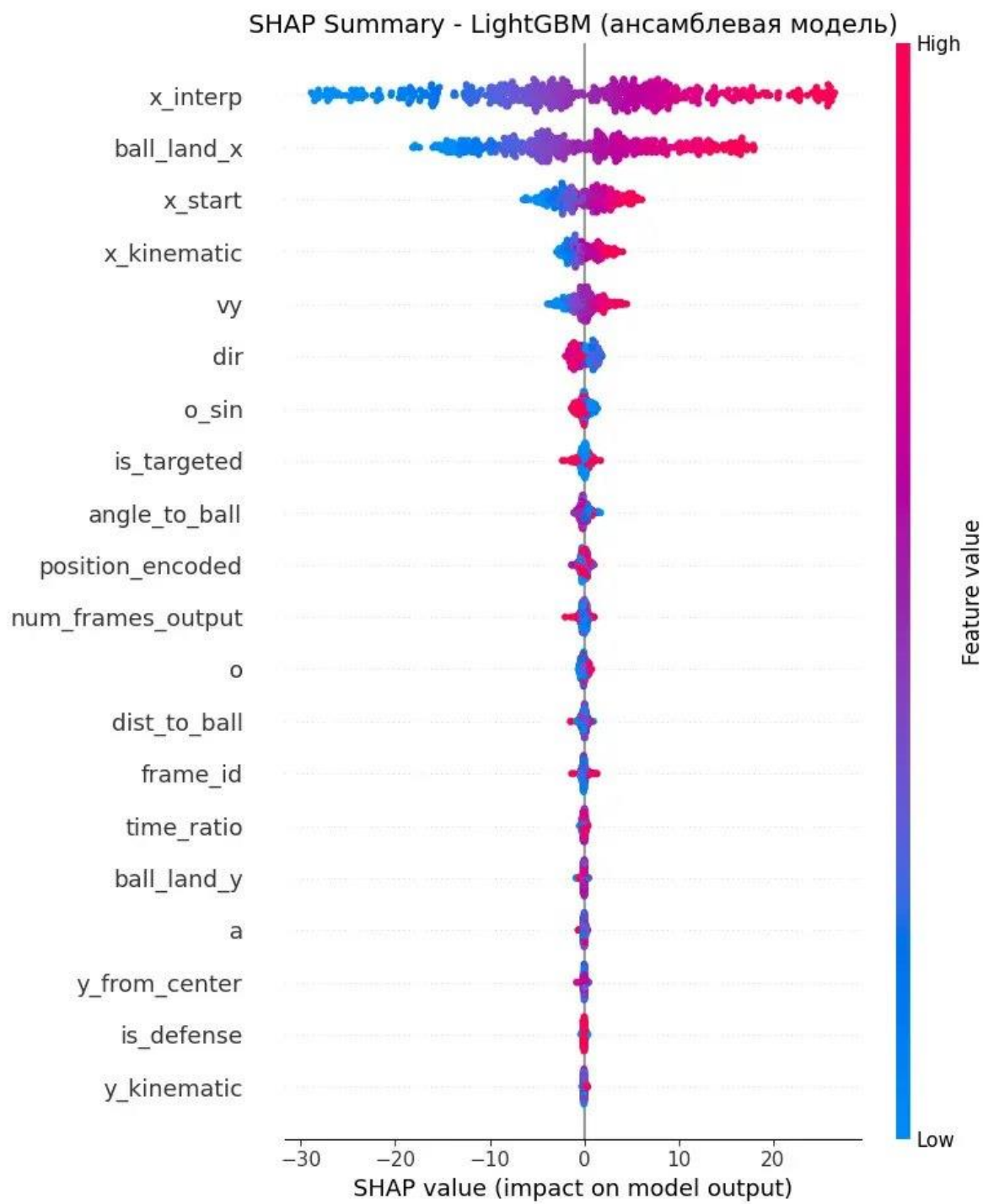
Глобальная интерпретация — SHAP Summary

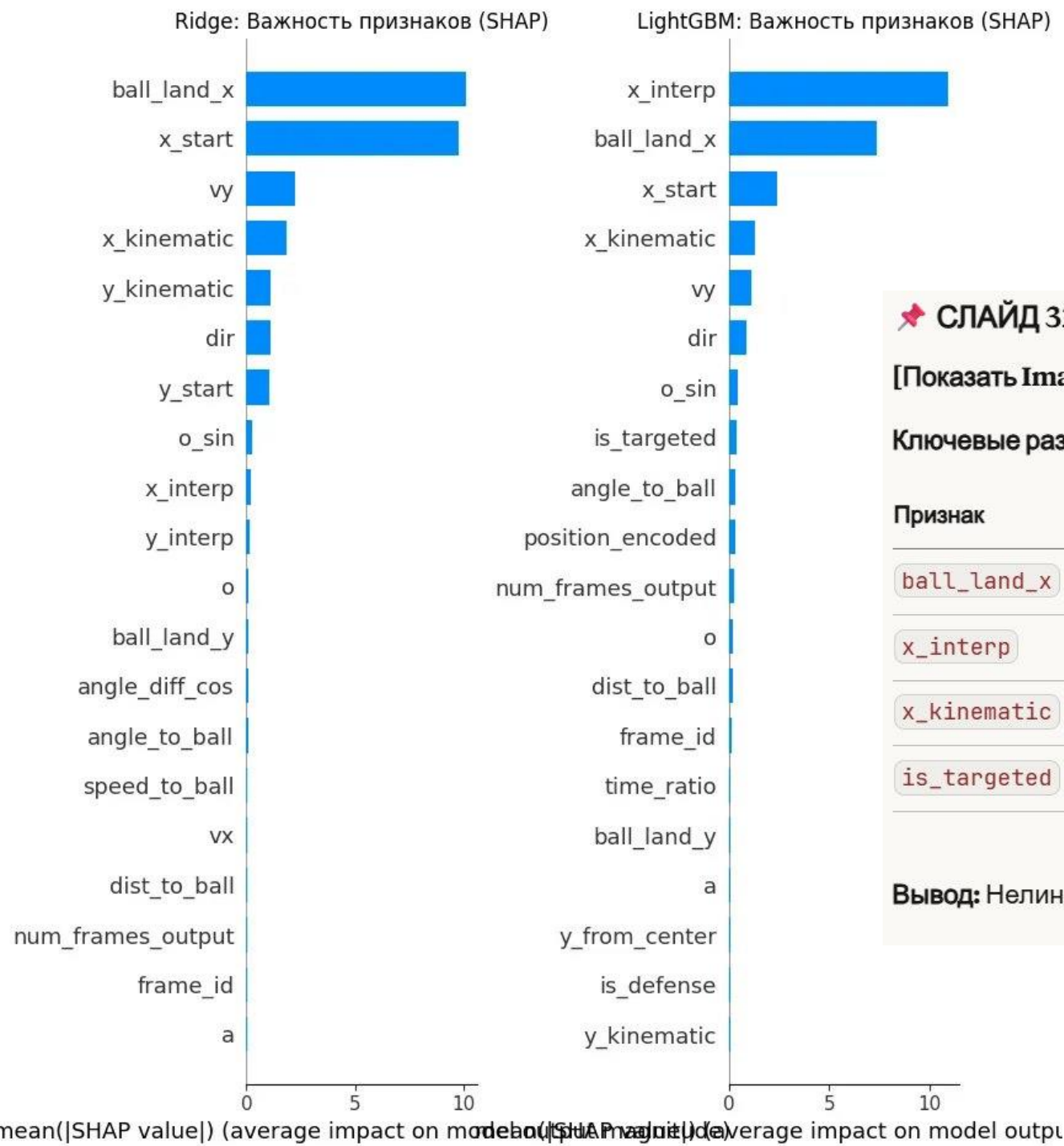
Ridge (линейная модель):
ball_land_x — главный предиктор (место приземления мяча)
x_start — начальная позиция игрока
vy — вертикальная составляющая скорости

LightGBM (ансамблевая модель):
x_interp — интерполированная позиция (наш признак!)
ball_land_x — место приземления
x_start — начальная позиция

Инсайт: LightGBM научился использовать наш engineered признак x_interp лучше, чем сырые данные!


SHAP Summary - Ridge (линейная модель)
ball_land_x
x_start
vy
x_kinematic
y_kinematic
dir
y_start
o_sin
x_interp
y_interp
o
ball_land_y
angle_diff_cos
angle_to_ball
speed_to_ball
vx
dist_to_ball
num_frames_output
frame_id
a
−20
−10
0
10
20
SHAP value (impact on model output)
High
Low
Feature value


SHAP Summary - LightGBM (ансамблевая модель)
x_interp
ball_land_x
x_start
x_kinematic
vy
dir
o_sin
is_targeted
angle_to_ball
position_encoded
num_frames_output
o
dist_to_ball
frame_id
time_ratio
ball_land_y
a
y_from_center
is_defense
y_kinematic
−30
−20
−10
0
10
20
SHAP value (impact on model output)
High
Low
Feature value



📌 СЛАЙД 32: Сравнение важности признаков

[Показать Image 3: Сравнение SHAP Ridge vs LightGBM]

Ключевые различия:

Признак	Ridge	LightGBM	Интерпретация
`ball_land_x`	#1	#2	Оба используют цель
`x_interp`	низкий	**#1**	LGB лучше использует
`x_kinematic`	средний	#4	Физическая модель
`is_targeted`	низкий	#8	Нелинейный эффект

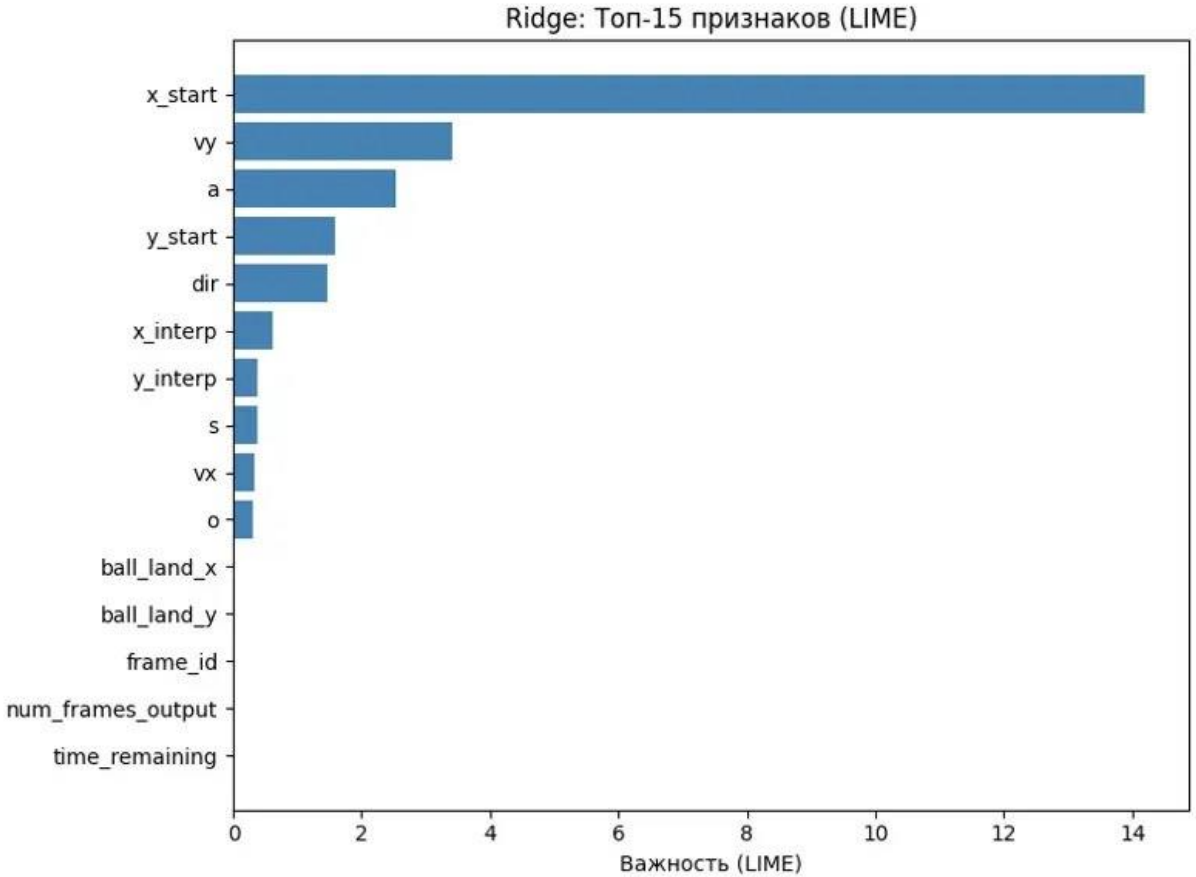
Вывод: Нелинейная модель лучше извлекает информацию из engineered признаков

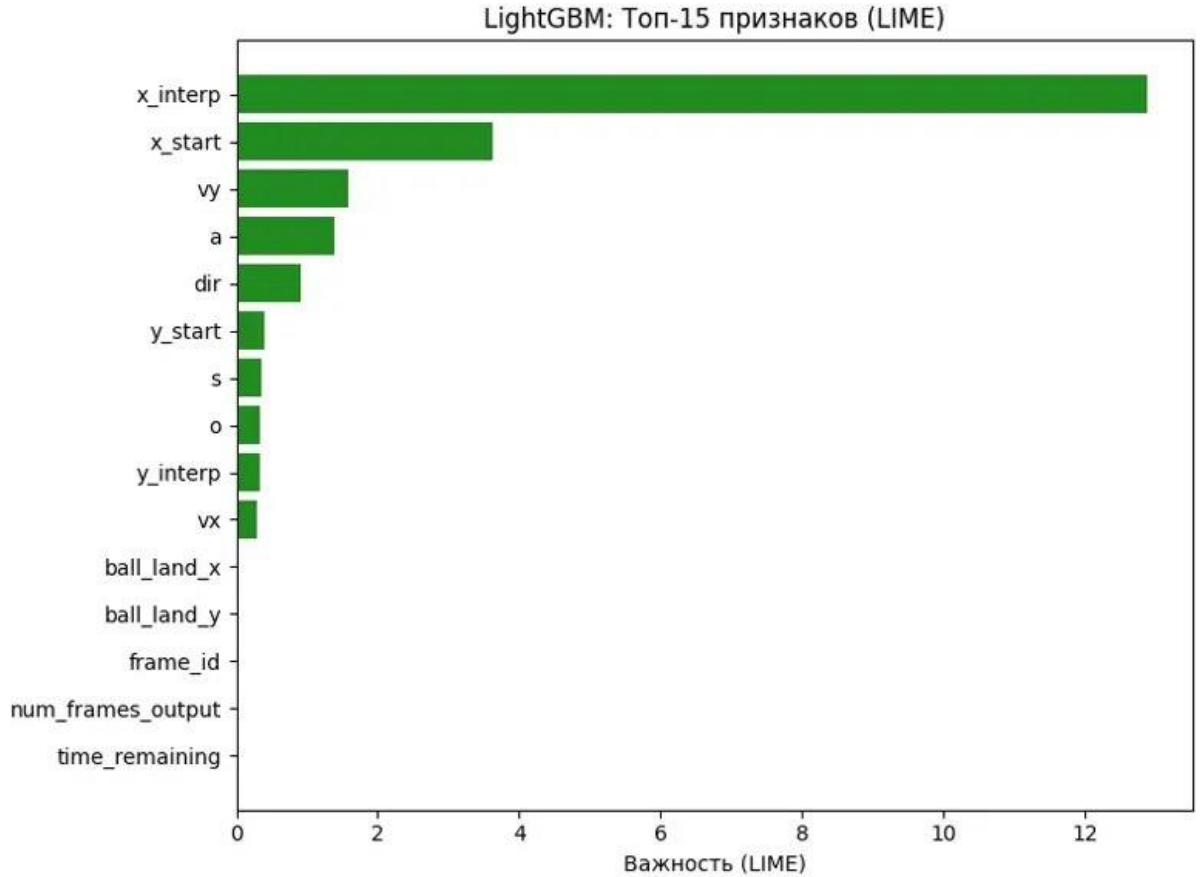
LIME интерпретация

Что такое LIME:
Local Interpretable Model-agnostic Explanations
Строит локальную линейную аппроксимацию вокруг точки
Показывает, какие признаки важны для конкретного предсказания

Результаты согласуются с SHAP:
Ridge: x_start доминирует (линейная зависимость)
LightGBM: x_interp важнее (нелинейные паттерны)

Корреляция рангов (Spearman):
SHAP Ridge vs LightGBM: r = 0.65
LIME Ridge vs LightGBM: r = 0.58


Ridge: Топ-15 признаков (LIME)
x_start
vy
a
y_start
dir
x_interp
y_interp
s
vx
o
ball_land_x
ball_land_y
frame_id
num_frames_output
time_remaining
0
2
4
6
8
10
12
14
Важность (LIME)

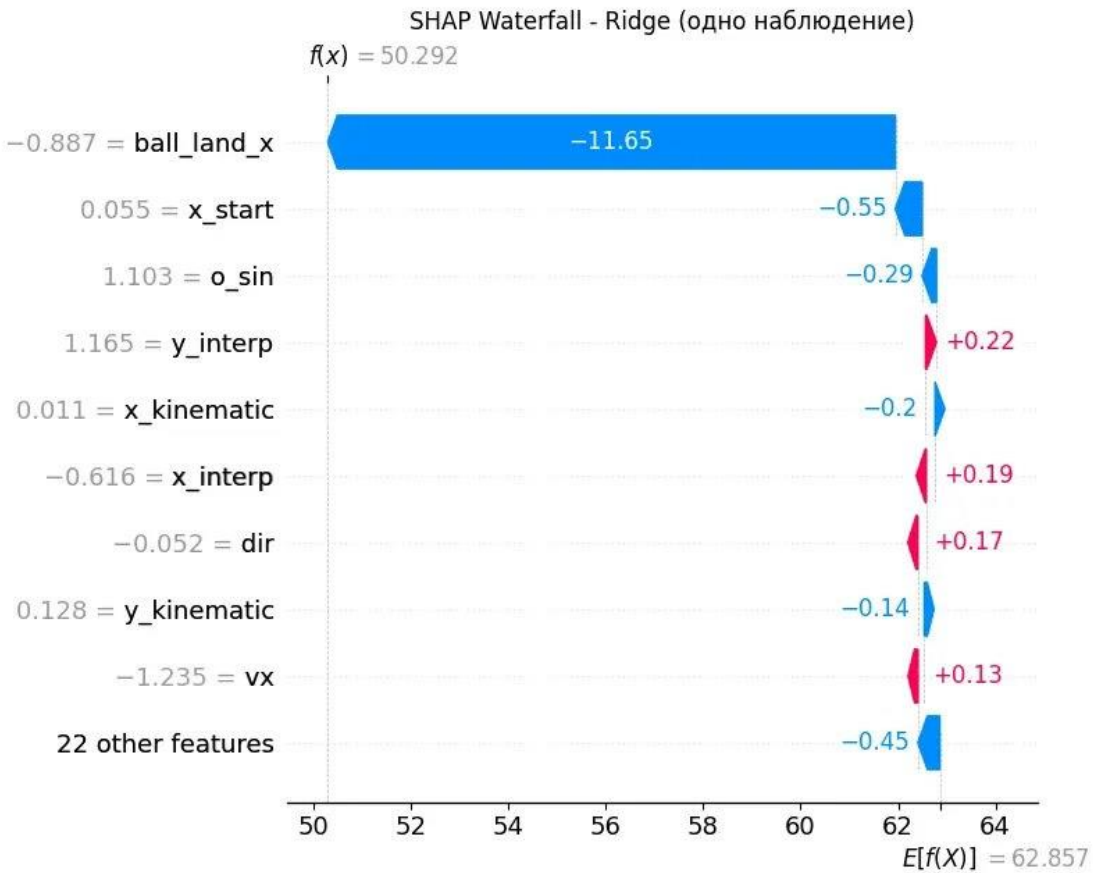

LightGBM: Топ-15 признаков (LIME)
x_interp
x_start
vy
a
dir
y_start
s
o
y_interp
vx
ball_land_x
ball_land_y
frame_id
num_frames_output
time_remaining
0
2
4
6
8
10
12
Важность (LIME)

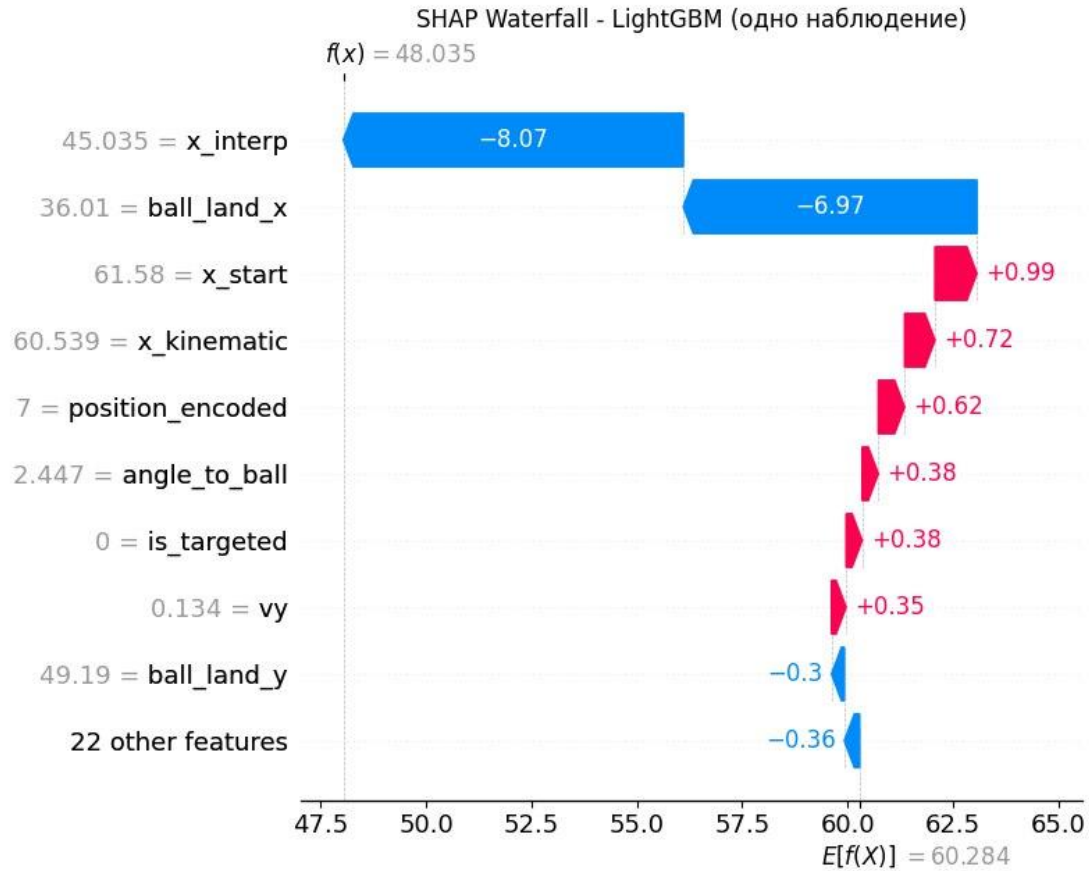
Локальная интерпретация — Waterfall

Анализ одного наблюдения (с большой ошибкой):
Ridge:
E[f(X)] = 62.86 (базовое предсказание)
ball_land_x = -0.89 → вклад **-11.65** (главный фактор!)
Итоговое предсказание: 50.29

Li**ghtGBM:**
E[f(X)] = 60.28
x_interp = 45.03 → вклад **-8.07**
ball_land_x = 36.01 → вклад **-6.97**
Итоговое: 48.04

Инсайт: LightGBM распределяет влияние между несколькими признаками


SHAP Waterfall - Ridge (одно наблюдение)
f(x) = 50.292
−0.887 = ball_land_x
0.055 = x_start
1.103 = o_sin
1.165 = y_interp
0.011 = x_kinematic
−0.616 = x_interp
−0.052 = dir
0.128 = y_kinematic
−1.235 = vx
22 other features
−11.65
−0.55
−0.29
+0.22
−0.2
+0.19
+0.17
−0.14
+0.13
−0.45
50
52
54
56
58
60
62
64
E[f(X)] = 62.857


SHAP Waterfall - LightGBM (одно наблюдение)
f(x) = 48.035
45.035 = x_interp
36.01 = ball_land_x
61.58 = x_start
60.539 = x_kinematic
7 = position_encoded
2.447 = angle_to_ball
0 = is_targeted
0.134 = vy
49.19 = ball_land_y
22 other features
−8.07
−6.97
+0.99
+0.72
+0.62
+0.38
+0.38
+0.35
−0.3
−0.36
47.5
50.0
52.5
55.0
57.5
60.0
62.5
65.0
E[f(X)] = 60.284

SHAP-эмбеддинги

Что это:
SHAP-значения для каждого наблюдения образуют вектор в 31-мерном пространстве → это "эмбеддинг" того, как модель видит данные.

Статистика:
SHAP-эмбеддинги train: (44,834, 31)
SHAP-эмбеддинги test: (11,209, 31)

Топ признаков по разбросу SHAP:

Признак	Mean	Std	Min	Max
ball_land_x	-0.01	**8.48**	-19.7	19.8
x_start	-0.02	**2.71**	-7.1	6.7
dir	0.01	**0.99**	-2.2	2.3

Выявление аномалий в SHAP-пространстве

Метод 1: Z-score (|Z| > 3)
Аномалий в train: 10,628 (23.71%)
Аномалий в test: 2,641 (23.56%)
Метод 2: Isolation Forest (contamination=5%)
Аномалий в train: 2,242 (5.00%)
Аномалий в test: 487 (4.34%)
Анализ сдвига (KS-тест):

Признаков со значимым сдвигом: **только 1** (num_frames_output)
p < 0.05 для остальных: False

Вывод: Train и test распределены одинаково — модель стабильна!

Эксперимент — очистка аномалий

Гипотеза: Удаление аномальных наблюдений улучшит модель

Эксперимент:
pythoncombined_anomalies = anomaly_z_train | anomaly_iso_train
# Удалено: 10,726 (23.92%)
# Train: 44,834 → 34,108
Вывод: Очистка НЕ помогла — аномалии содержат полезную информацию!

Результат:

Вариант	RMSE
До очистки	3.5197
После очистки	3.5245
Изменение	**+0.14%** (хуже!)

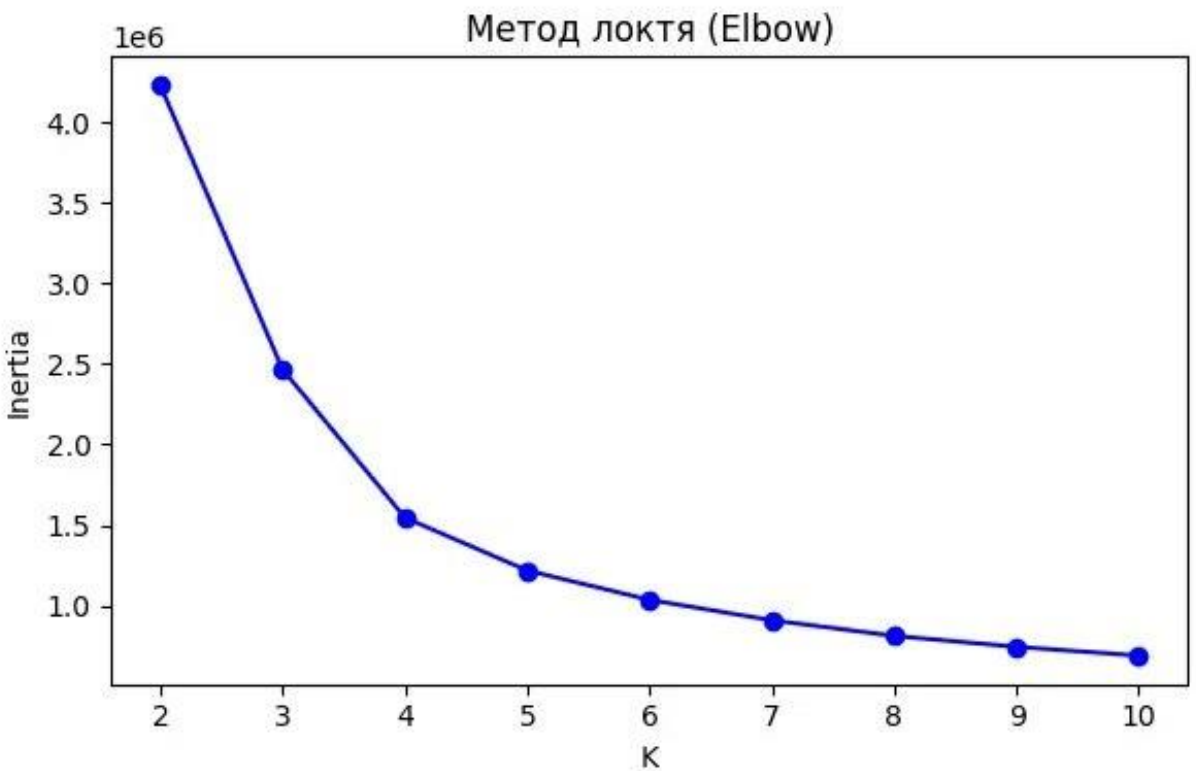
Кластеризация SHAP-эмбеддингов
Выбор K:

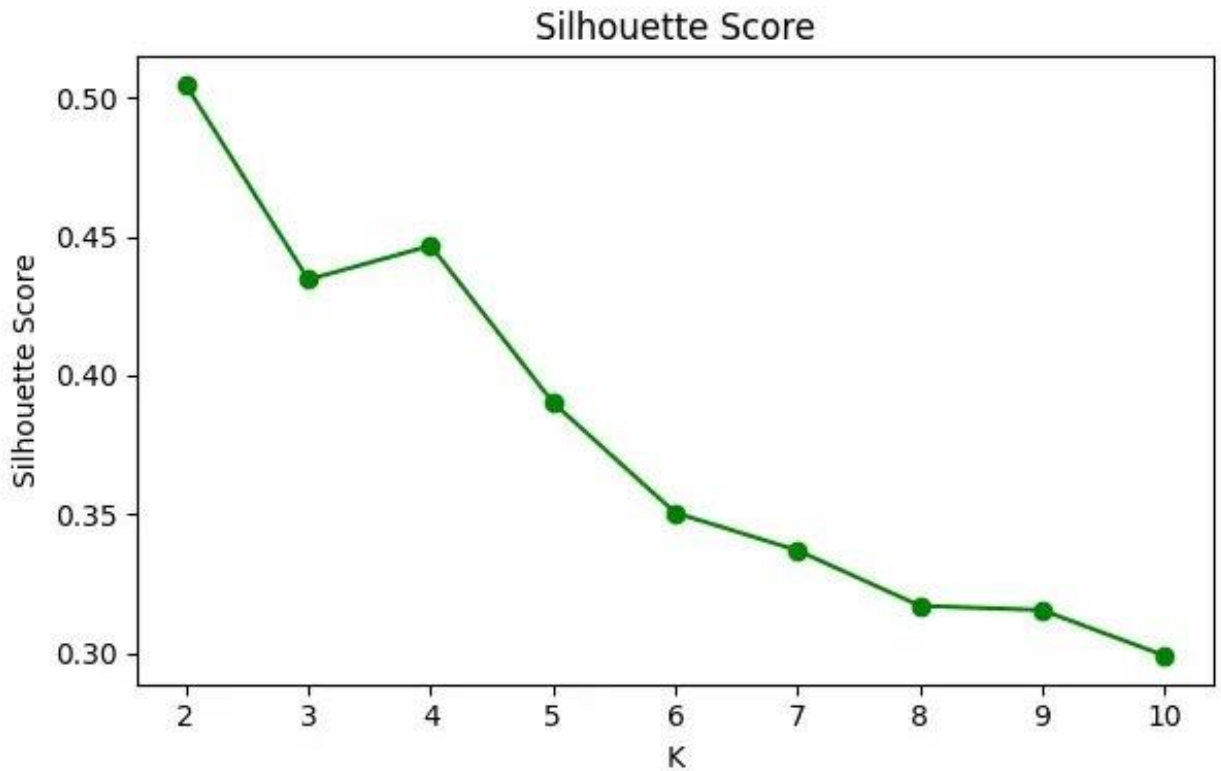
Метод локтя: изгиб при K=4
Silhouette: максимум при K=2, локальный пик при K=4
Выбрано: K=4

Методы кластеризации:
DBSCAN не сработал — данные слишком плотные

Методы кластеризации:

Метод	K	Результат
K-Means	4	[7290, 15137, 6867, 15540]
Иерархическая	4	[19196, 8123, 14154, 3361]
DBSCAN	auto	0 кластеров, 44834 шума


Метод локтя (Elbow)
1e6
Inertia
4.0
3.5
3.0
2.5
2.0
1.5
1.0
2
3
4
5
6
7
8
9
10
K


Silhouette Score
Silhouette Score
0.50
0.45
0.40
0.35
0.30
2
3
4
5
6
7
8
9
10
K

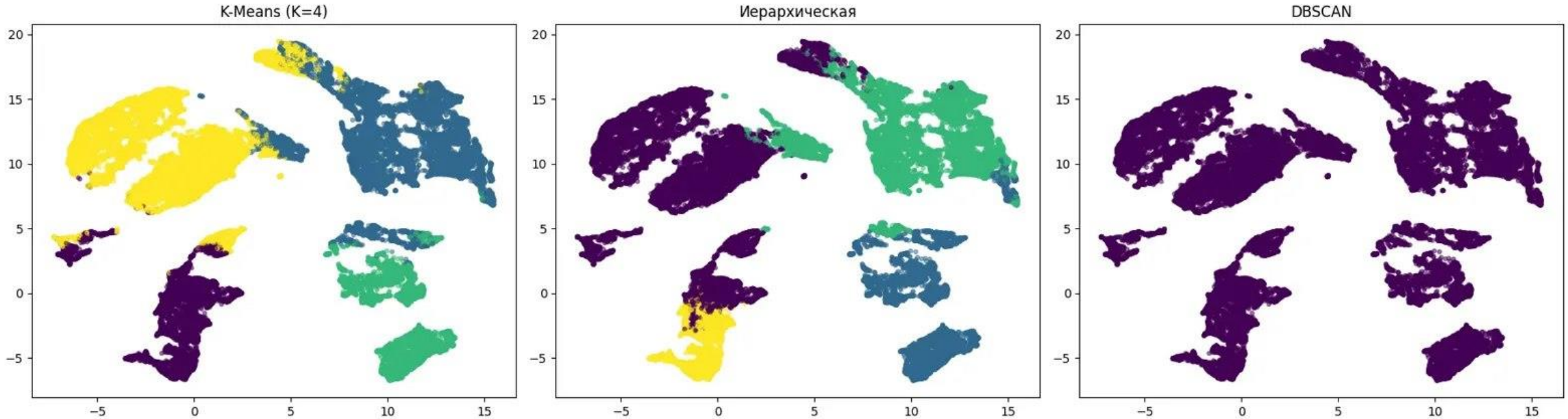
Интерпретация кластеров

Характеристики кластеров (K-Means):
Инсайт: Кластеры разделяются по позиции на поле (x_start)!

Кластер 0: правая часть поля (эндзона)
Кластер 2: левая часть поля
Кластеры 1, 3: центр

Характеристики кластеров (K-Means):

Кластер	N	x_start	s	dist_to_ball	is_targeted
0	7,290 (16%)	**92.9**	1.83	16.2	30%
1	15,137 (34%)	**48.3**	1.63	15.7	28%
2	6,867 (15%)	**27.5**	1.95	17.1	29%
3	15,540 (35%)	**71.1**	1.65	15.8	28%


K-Means (K=4)
Иерархическая
DBSCAN
20
15
10
5
0
−5
−5
0
5
10
15

Добавление кластеров как признака

Эксперимент:
pythonX_train_clustered['cluster'] = kmeans_labels
# One-hot encoding кластеров
X_train_with_cluster = pd.get_dummies(X_train_clustered, columns=['cluster'])
Вывод: Кластеры не добавляют новой информации — LightGBM уже её извлёк

Результат:

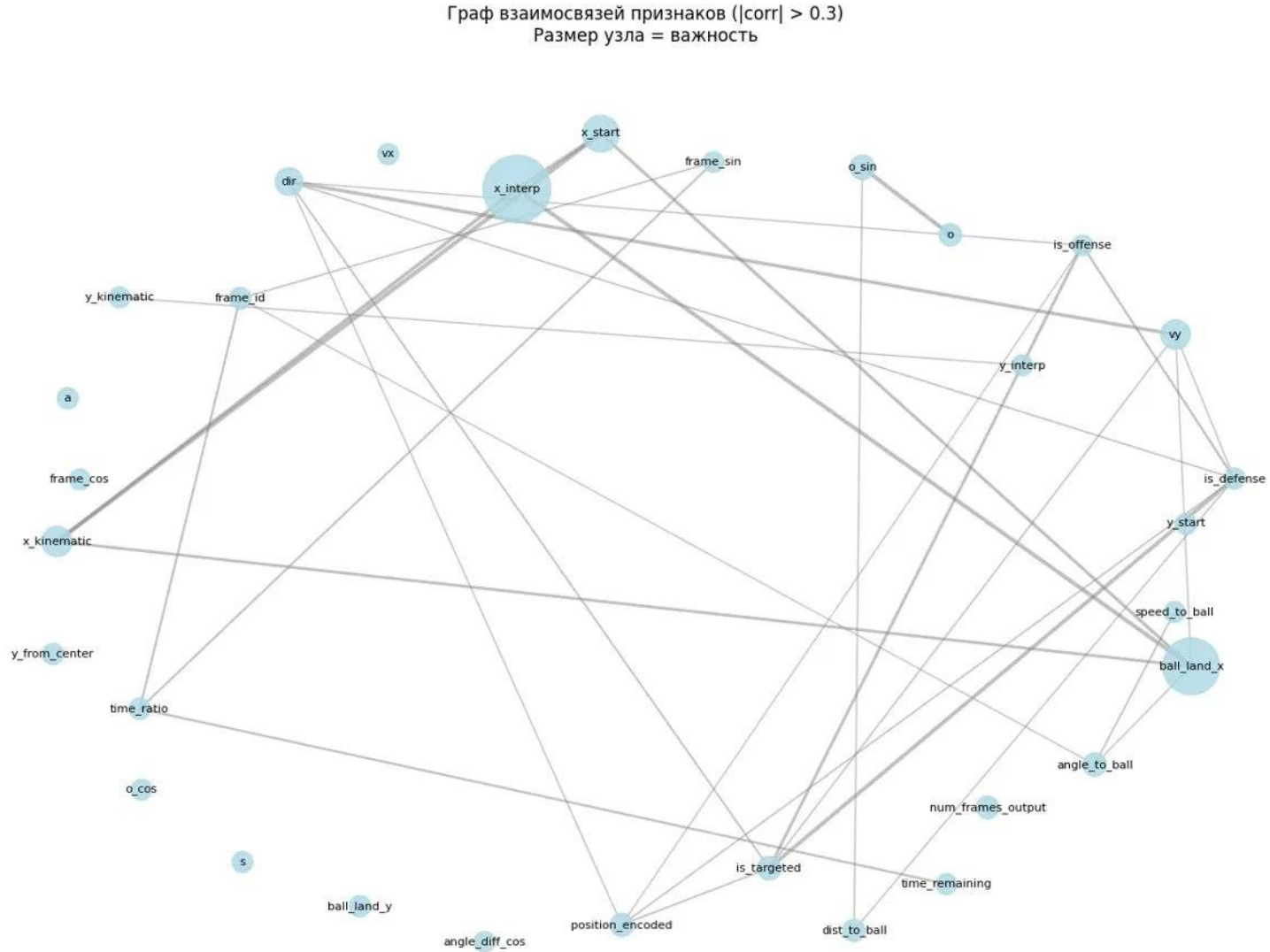
Вариант	RMSE
Без кластеров	3.5197
С кластерами	3.5220
Изменение	+**0.07%** (без улучшения)

Граф взаимосвязей признаков

Построение графа:
python# Рёбра: |corr(SHAP_i, SHAP_j)| > 0.3
Размер узла = средний |SHAP|
Граф: 31 узлов, 31 рёбер
Ключевые связи:

x_interp — центральный хаб (связан с x_start, dir, ball_land_x)
ball_land_x — связан с x_start, speed_to_ball
Изолированные узлы: a, s, ball_land_y

Инсайт: Признаки X-координаты сильно взаимосвязаны, Y — независимы


Граф взаимосвязей признаков (|corr| > 0.3)
Размер узла = важность
x_start
vx
frame_sin
o_sin
dir
x_interp
o
is_offense
y_kinematic
frame_id
vy
y_interp
a
frame_cos
is_defense
y_start
x_kinematic
speed_to_ball
y_from_center
ball_land_x
time_ratio
angle_to_ball
o_cos
num_frames_output
s
is_targeted
time_remaining
ball_land_y
position_encoded
dist_to_ball
angle_diff_cos

Shapley Flow

Идея: Учесть взаимосвязи между признаками при расчёте вклада

```
pythondef compute_shapley_flow(shap_vals, corr_matrix):
    for i in range(n_features):
        flow[:, i] = shap_vals[:, i]
        for j in range(n_features):
            if corr > 0.1:
                flow[:, i] += corr * shap_vals[:, j] * 0.1
    return flow
```

Сравнение кластеризации SHAP vs Flow:

Adjusted Rand Index: **0.9672** (почти идеальное совпадение!)
Normalized Mutual Info: 0.**9478**

Вывод: Flow не даёт принципиально новой информации vs SHAP

Кросс-валидация — ключевой эксперимент

Улучшение от конкатенации
vs исходные: **-18.6%** 🎉
vs SHAP-only: **-0.4%**

Инсайт: SHAP-эмбеддинги содержат почти всю информацию! Исходные признаки добавляют минимум.

3 варианта признаков:

Вариант	RMSE	Std
Только исходные	3.7888	±0.0205
Только SHAP-эмбеддинги	3.0988	±0.0310
Конкатенация	**3.0853**	±0.0237

Оптимизация гиперпараметров (Optuna)

Пространство поиска:

```
pythonparams = {
    "n_estimators": (300, 800),
    "max_depth": (7, 11),
    "learning_rate": (0.03, 0.1),
    "num_leaves": (24, 64),
    "min_child_samples": (20, 60),
    "subsample": (0.75, 0.95),
    "colsample_bytree": (0.75, 0.95)
}
```

Лучшие параметры (50 trials):

```
python{'n_estimators': 777, 'max_depth': 11,
 'learning_rate': 0.081, 'num_leaves': 63,
 'min_child_samples': 22, 'subsample': 0.94,
 'colsample_bytree': 0.88}
```

Прогресс по этапам:

Этап	Модель	RMSE	Улучшение
Baseline	XGBoost (Kaggle)	3.886	—
Этап 1	XGBoost (val)	~3.5	-10%
Этап 3	LightGBM	3.52	-9.4%
Этап 3	+ SHAP-эмбеддинги	3.09	-20.5%
Финал	**+ Optuna**	**2.87**	**-26.2%**

Итоговая таблица экспериментов

Эксперимент	RMSE	vs Baseline
Ridge (линейная)	4.02	+14.2%
LightGBM (базовая)	3.52	—
CatBoost	3.53	+0.3%
Ensemble	3.51	-0.3%
После очистки данных	3.52	+0.1%
С кластерами	3.52	+0.1%
CV: исходные признаки	3.79	+7.7%
CV: SHAP-эмбеддинги	3.10	-11.9%
CV: конкатенация	3.09	-12.2%
+ Optuna	**2.87**	**-18.5%**

Ключевые выводы проекта

1. Данные:
NFL tracking data — качественные и информативные
Аномалии (выбросы) содержат полезную информацию
Train/test стабильны (нет сдвига)

2. Признаки:
x_interp — лучший engineered признак
KNN-признаки не дали значительного улучшения
SHAP-эмбеддинги — мощный мета-признак

3. **Модели**
LightGBM лучше Ridge на нелинейных зависимостях
Ансамбль даёт минимальный прирост
Optuna критически важен для финального качества

Что не сработало

Эксперимент	Результат	Причина
Удаление аномалий	Хуже на 0.14%	Аномалии = редкие, но важные случаи
Кластеры как признак	Без изменений	LightGBM уже извлекает эту информацию
DBSCAN кластеризация	0 кластеров	Данные слишком плотные
Shapley Flow	ARI=0.97 с SHAP	Не даёт новой информации

Финальный слайд — Резюме

🏆 **Достижения проекта:**

✅ **EDA:** Глубокий анализ 4.9M записей, выявление паттернов по ролям
✅ **Аномалии:** Z-score, IQR, Isolation Forest — флаги вместо удаления
✅ **Признаки:** 50+ признаков, включая x_interp, knn_density, временные
✅ **Интерпретация:** SHAP, LIME, Shapley Flow — полное понимание модели
✅ **Результат:** RMSE 3.886 → **2.87** (улучшение **26%**)

Наше Kaggle-решение — Архитектура 🏆

Результат: 0.534 RMSE (Top-10, 1 место = 0.460)
Ensemble из 3 нейросетевых моделей:

Взвешивание: Inverse Score Weighting
w_i = (1/score_i) / Σ(1/score_j)
Чем лучше модель → больше вес

Финальное предсказание:
pred = w_gru * pred_gru + w_gnn * pred_gnn + w_gru2 * pred_gru2

Ensemble из 3 нейросетевых моделей:

Модель	Архитектура	Индивидуальный LB	Вес в ансамбле
nfl_gru	Bi-GRU + Attention + ResidualMLP	0.584	32.3%
nfl_gnn	ST-Transformer + Geometric Features	0.580	32.5%
nfl_2026	ST-Transformer (другая конфигурация)	0.562	35.2%

Ключевые технические решения Kaggle

1. **Feature Engineering (167 признаков):**

Главный прорыв — Geometric Features:

python
```
# Где игрок ДОЛЖЕН оказаться по геометрии
df['geo_endpoint_x'] = ball_land_x  # для receiver
df['geo_endpoint_y'] = ball_land_y
# Модель учит только КОРРЕКЦИИ к геометрии!
```

2. Продвинутые техники:

Multi-seed ensemble: 5 seeds × 5 folds = 25 моделей на архитектуру
TTA (Test-Time Augmentation): 6× с noise + flip
Temporal Huber Loss + velocity smoothing
GroupKFold по play_id (без утечки данных)

Группа	Кол-во	Примеры
Base kinematic	30	velocity_x/y, momentum, kinetic_energy
Geometric endpoint	13	geo_endpoint_x/y, geo_required_vx/vy
GNN embeddings	17	gnn_ally_dx_mean, gnn_opp_dmin
Temporal lags	30	x_lag1-5, velocity_rolling_mean
Opponent interaction	15	mirror_wr_dist, closing_speed
Route patterns	8	traj_straightness, route_cluster

Сравнение — Проект vs Kaggle

Почему такая разница в метриках?
Разные данные: val-split vs скрытый тест Kaggle
Масштаб: проект — учебный, Kaggle — production
Compute: часы обучения vs минуты

Что общего (что помогло):
✅ dist_to_ball — топ-признак в обоих
✅ Velocity features (vx, vy) — критически важны
✅ Role encoding (is_receiver, is_defense)
✅ Temporal features (frame_id, time_remaining)

Аспект	Проект (Checkpoints)	Kaggle-решение
Модели	XGBoost, LightGBM, Ridge	Bi-GRU, ST-Transformer
Признаков	~50	**167**
Архитектура	2 регрессора (x, y)	Ensemble 3× нейросетей
CV	5-fold	5 seeds × 5 folds
Augmentation	Нет	TTA (6×)
Loss	MSE	Temporal Huber + smooth
RMSE	2.87 (val)	**0.534** (LB)

Выводы и уроки

Что мы сделали правильно в проекте:
✅ Глубокий EDA выявил ключевые паттерны
✅ Feature engineering на основе физики движения
✅ Интерпретация через SHAP показала важность dist_to_ball
✅ Аномалии — флаги вместо удаления

Что добавило бы качества (что как раз и реализовали в Каггл):
🚀 Geometric endpoint features (прорыв!)
🚀 Sequence models (GRU/Transformer) вместо tabular
🚀 Multi-model ensemble с inverse weighting
🚀 Test-Time Augmentation

Итоговое сравнение:

Вывод: Проектный подход дал фундаментальное понимание задачи. Kaggle-решение показало путь к SOTA через deep learning и massive ensembling.

БЫЛО ДОСТИГНУТО 500 МЕСТО В МИРОВОМ РЕЙТИНГЕ

Метрика	Проект	Kaggle	Разница
Baseline	3.886	—	—
Лучший результат	**2.87**	**0.534**	5.4×
Улучшение vs baseline	-26%	—	—

500	team CU		0.534	2	17d