

# 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

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

# 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, у на поле

*Датасет полностью соответствует требованиям проекта*

# EDA — Распределение ролей игроков

## Ключевые наблюдения:

**Defensive Coverage:** 155,397 записей (доминирует) — защитники активно реагируют на пас

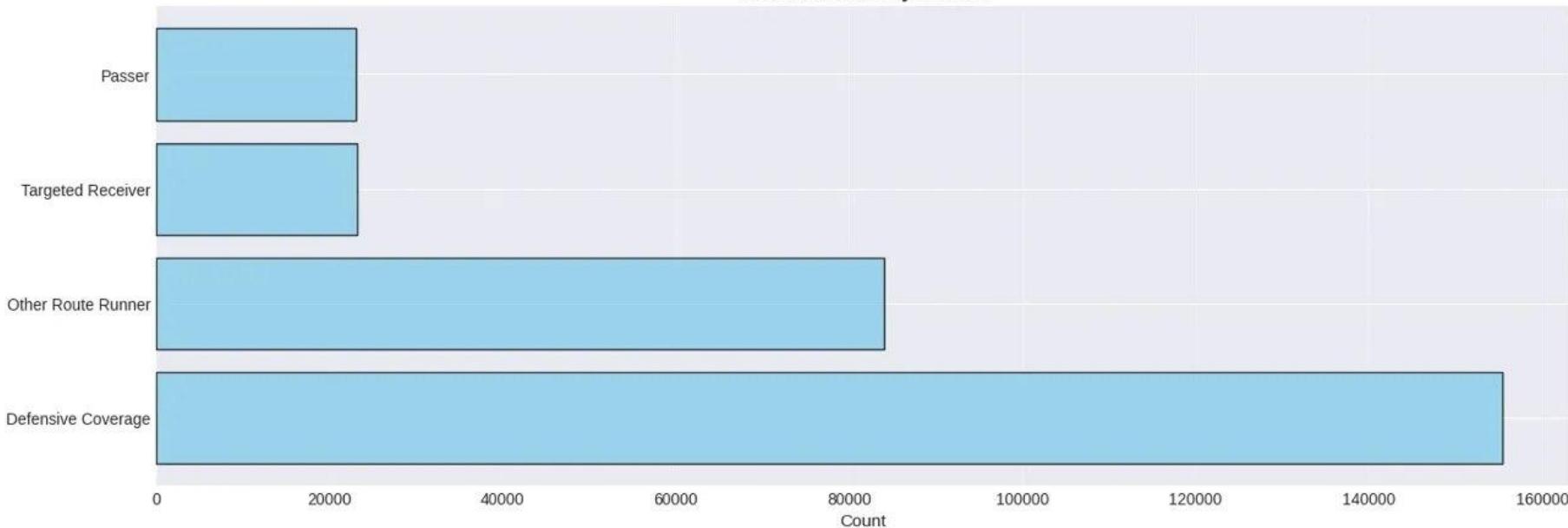
**Other Route Runner:** 84,063 — другие принимающие, создающие отвлечение

**Targeted Receiver:** 23,151 — целевой принимающий (главный объект предсказания)

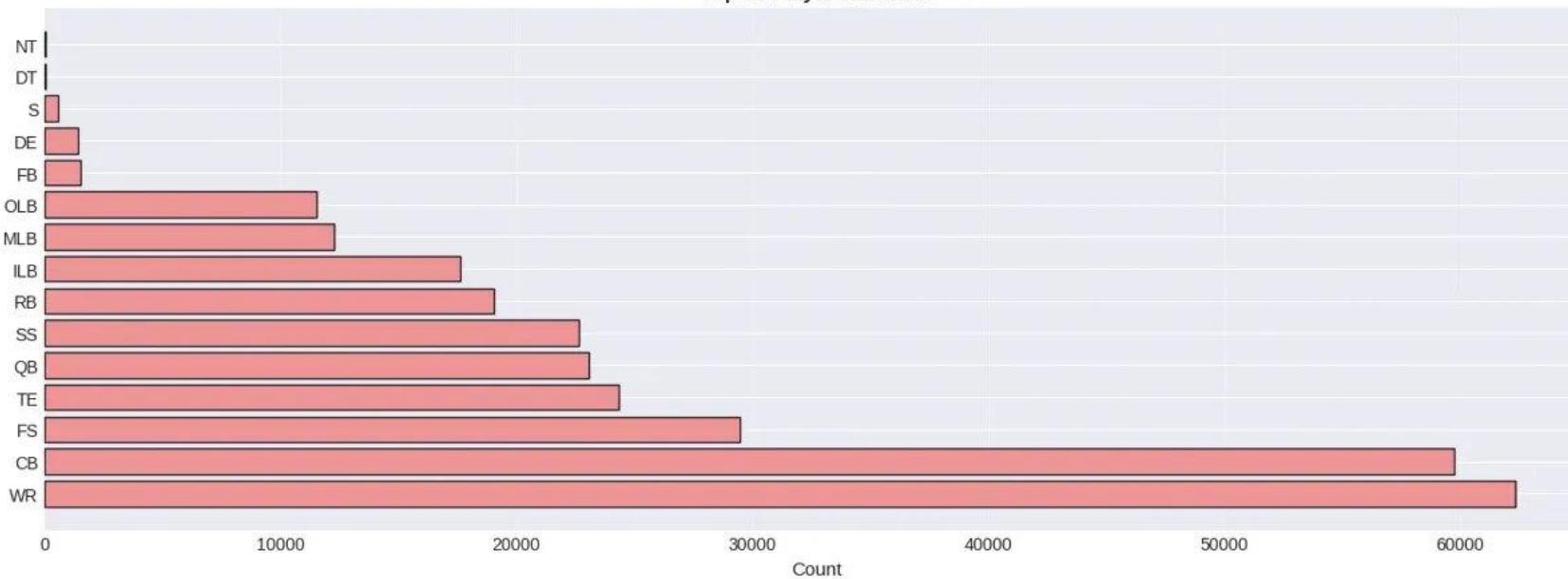
**Passer:** 23,103 — квотербек, бросающий мяч

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

**Distribution of Player Roles**



**Top 15 Player Positions**



# EDA — Скорость и ускорение

## Статистика скоростей:

Средняя скорость: **3.04 ярда/сек**

Распределение: правосторонний скос (много медленных, мало быстрых)

Максимум: ~12 ярдов/сек (спринт)

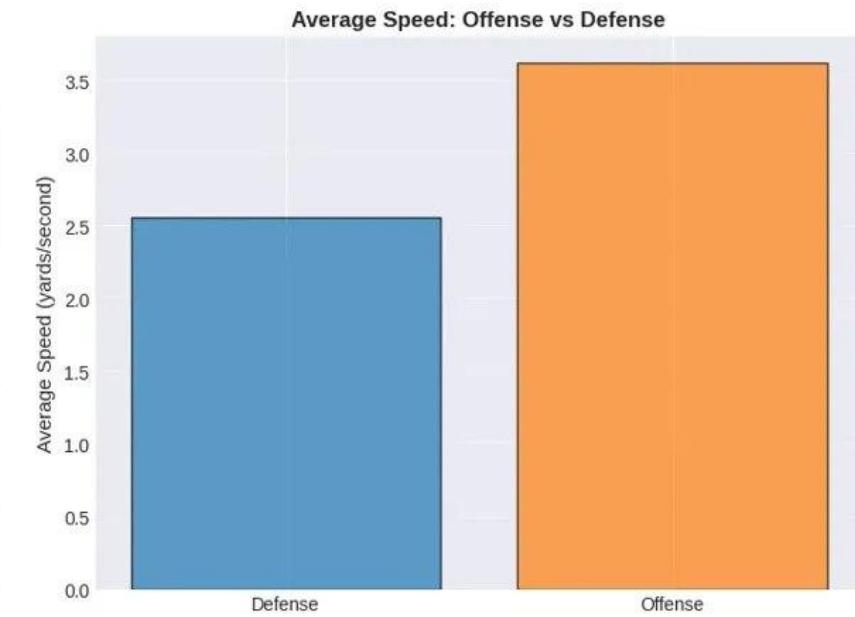
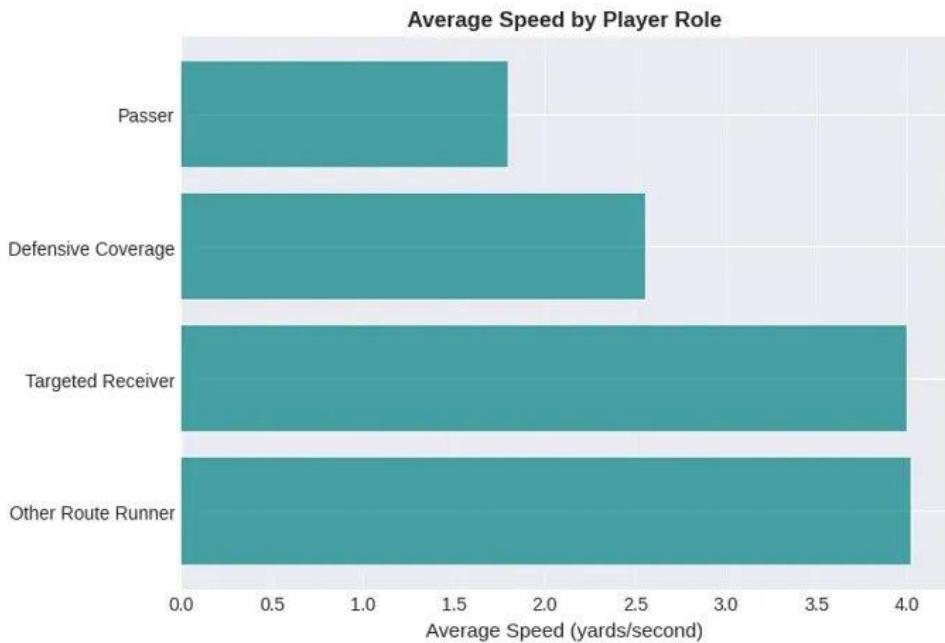
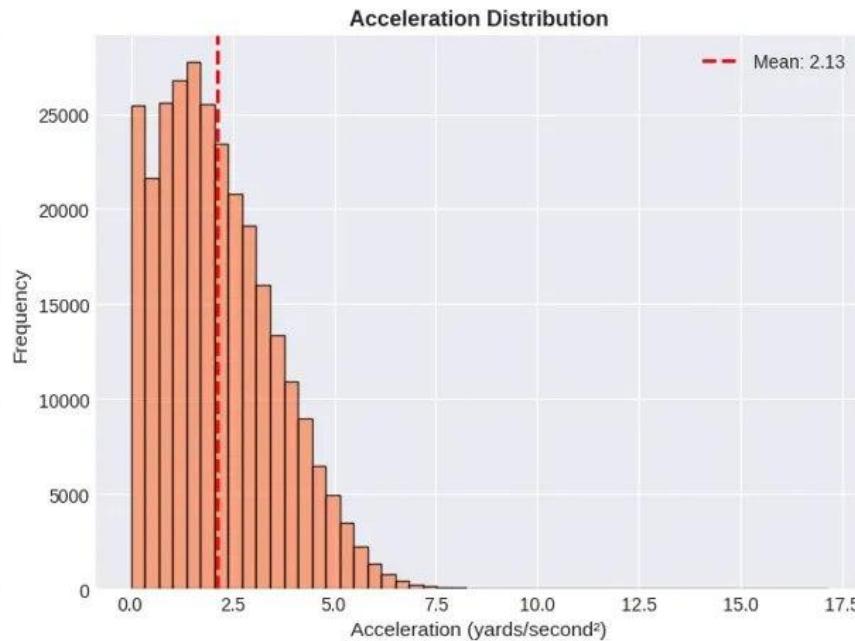
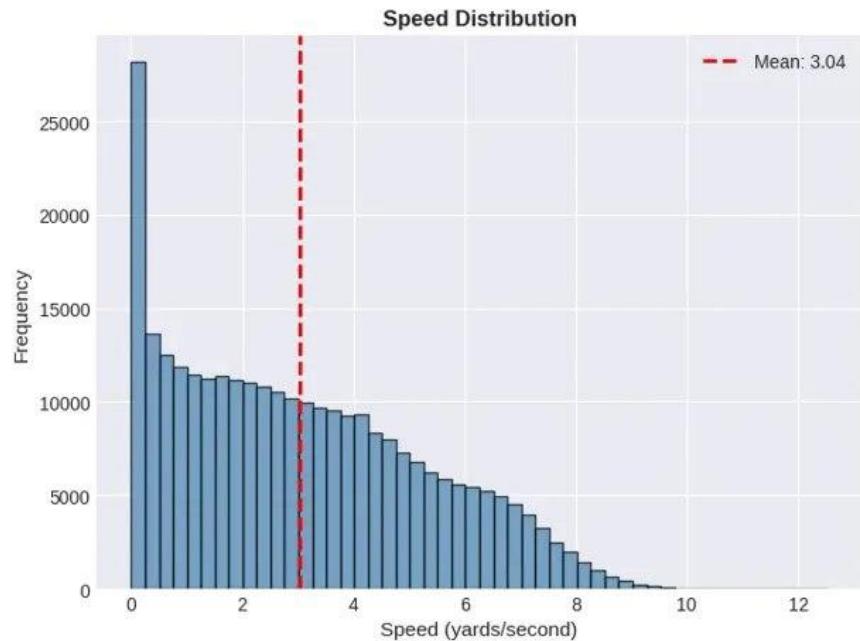
## По ролям игроков:

**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



# 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)

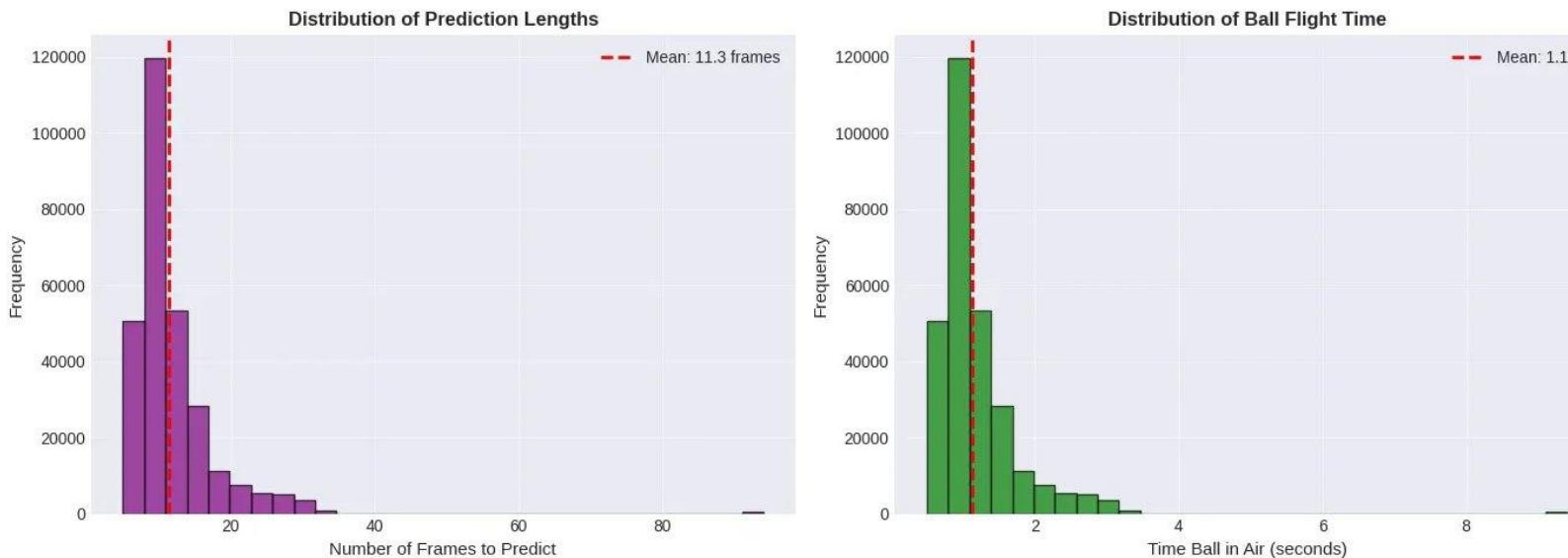
75% розыгрышей: ≤13 кадров

## Инсайт:

Большинство пасов — короткие (quick passes)

Модель должна хорошо работать на коротких последовательностях

Длинные пасы (>30 кадров) — редкие, но сложные случаи



# EDA — Позиции на поле

## Наблюдения:

Игроки концентрируются в центре поля ( $y \approx 26.65$  ярдов)

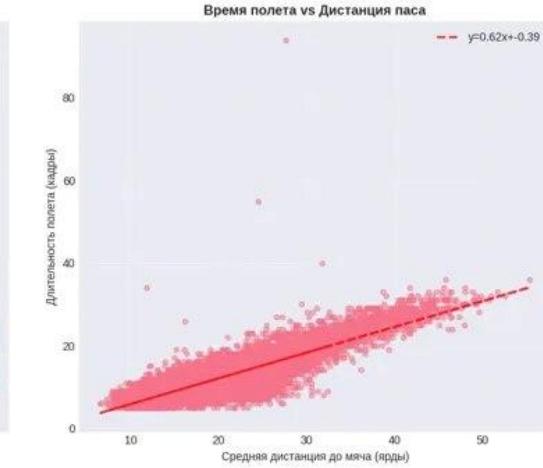
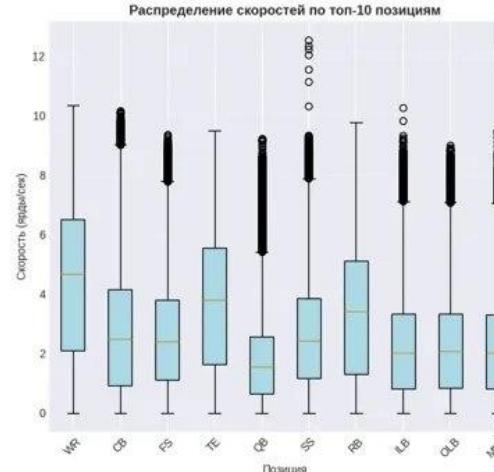
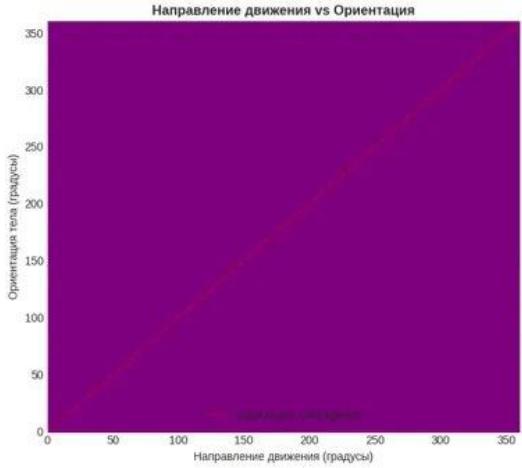
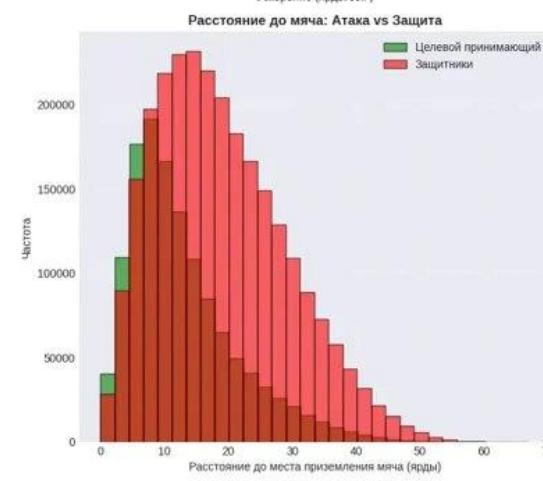
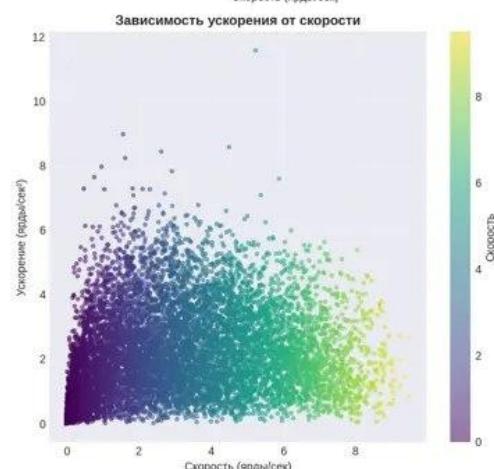
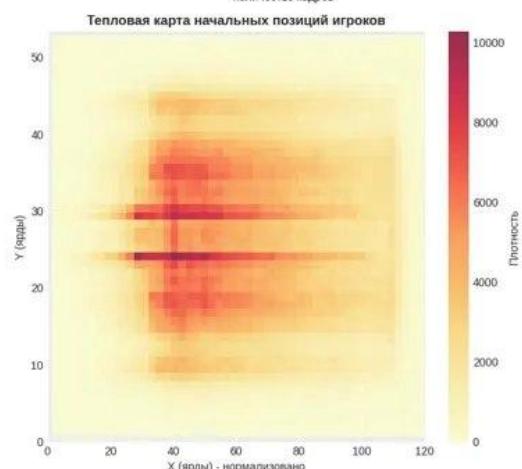
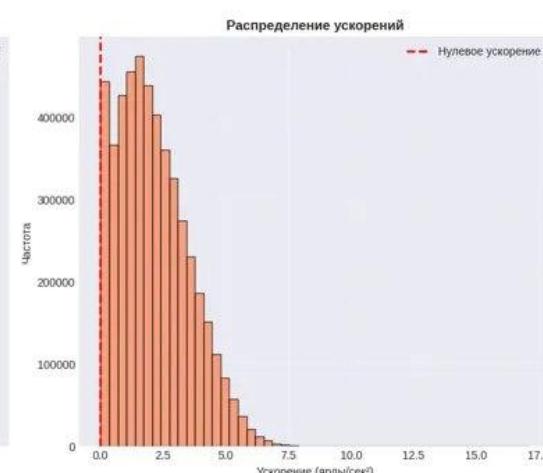
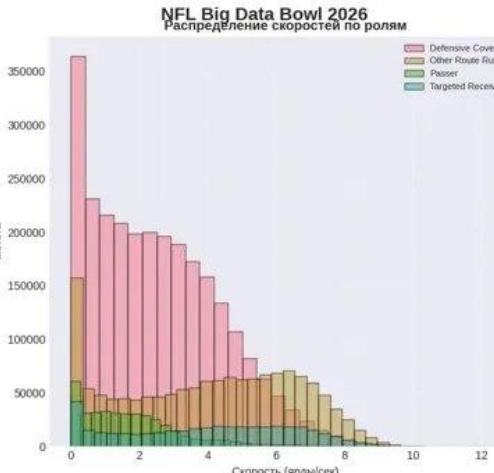
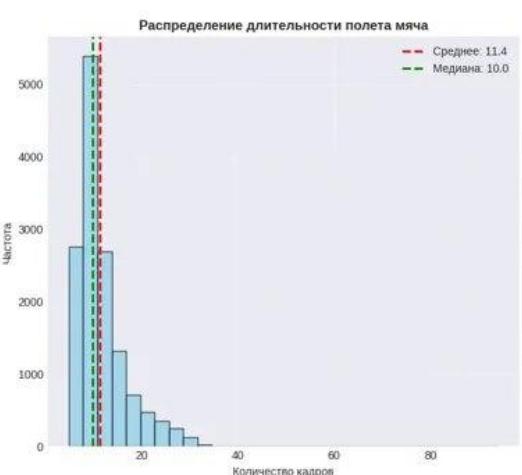
Горизонтальные "полосы" — типичные построения до паса  
x координата варьируется широко (зависит от позиции на поле)

## Корреляции:

$x \leftrightarrow ball\_land\_x: 0.88$  (сильная связь!)

$x \leftrightarrow absolute\_yardline: 0.94$

Скорость и ускорение слабо коррелируют с позицией



# EDA — Расстояние до мяча

**Средняя дистанция до места приземления:**

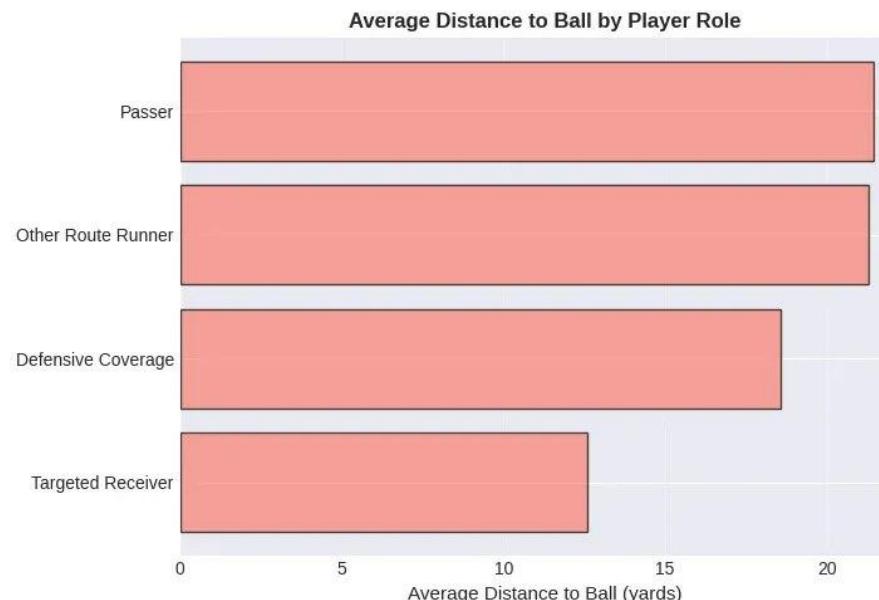
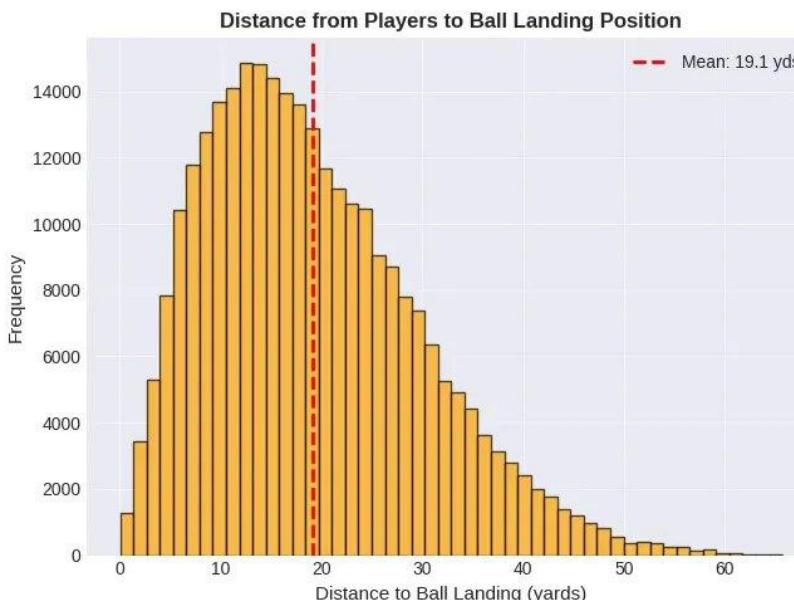
**Targeted Receiver:** ~12 ярдов (ближе всех к мячу)

**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!



# Анализ физики движения

## Проблема выбросов:

Distance Ratio (факт/теория): **77** для ресиверов, **55** для защитников

Ожидаемое значение: 1.0–1.2

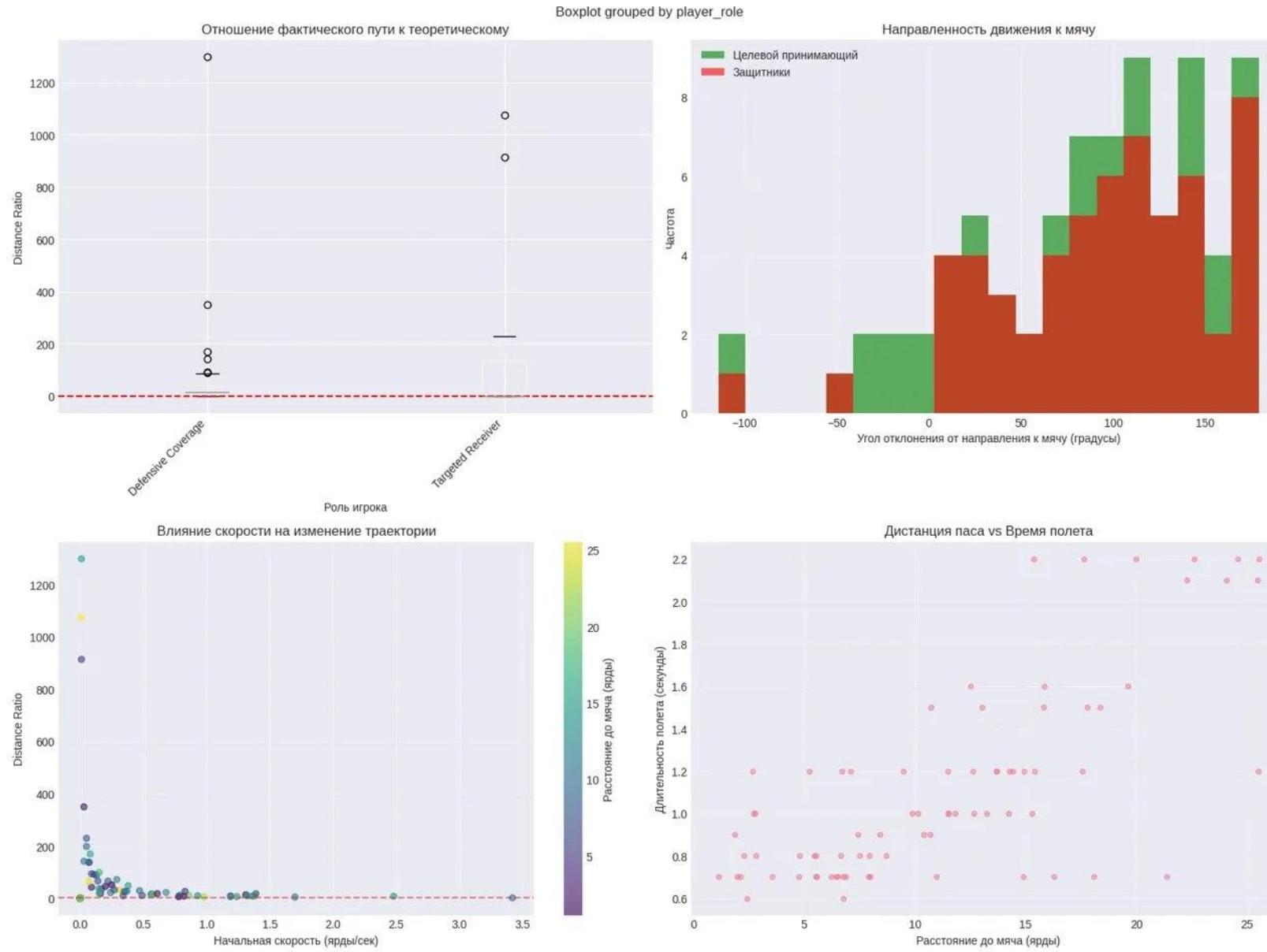
Причина: деление на малые скорости (игроки почти неподвижны)

## Направленность к мячу:

Целевой принимающий: отклонение **87°** от направления к мячу

Задачники: отклонение 94°

**Вывод:** Нужна фильтрация выбросов по скорости перед обучением



# 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 у

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
<b>Kaggle RMSE</b>	<b>3.886 ярда</b>
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

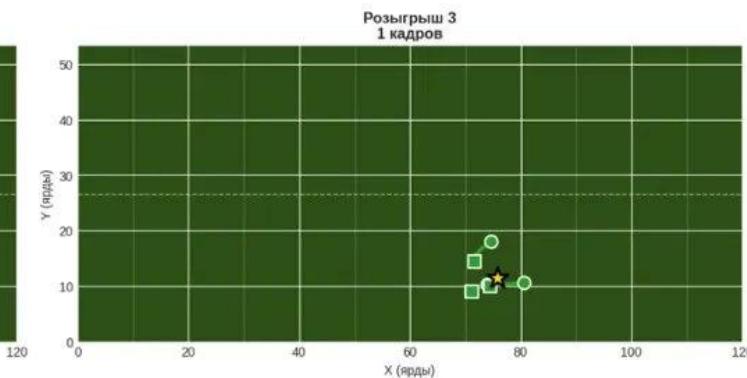
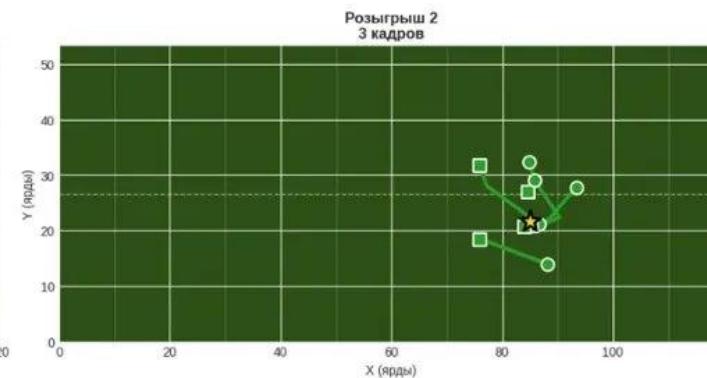
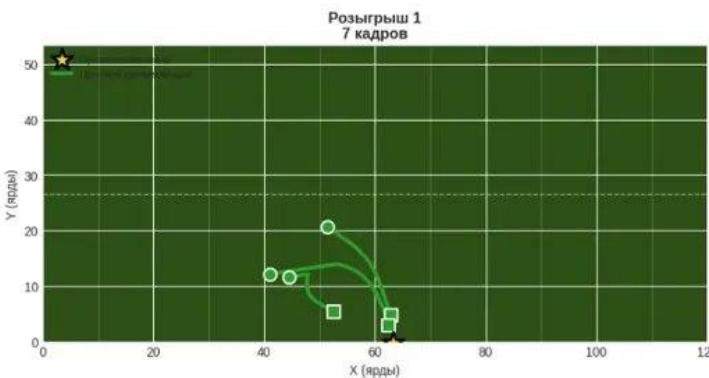
Розыгрыш 3 (1 кадр): мгновенный пас

## 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
<b>Baseline XGB</b>	<b>40 признаков</b>	<b>3.886</b>
→ 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)

```
python df['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)

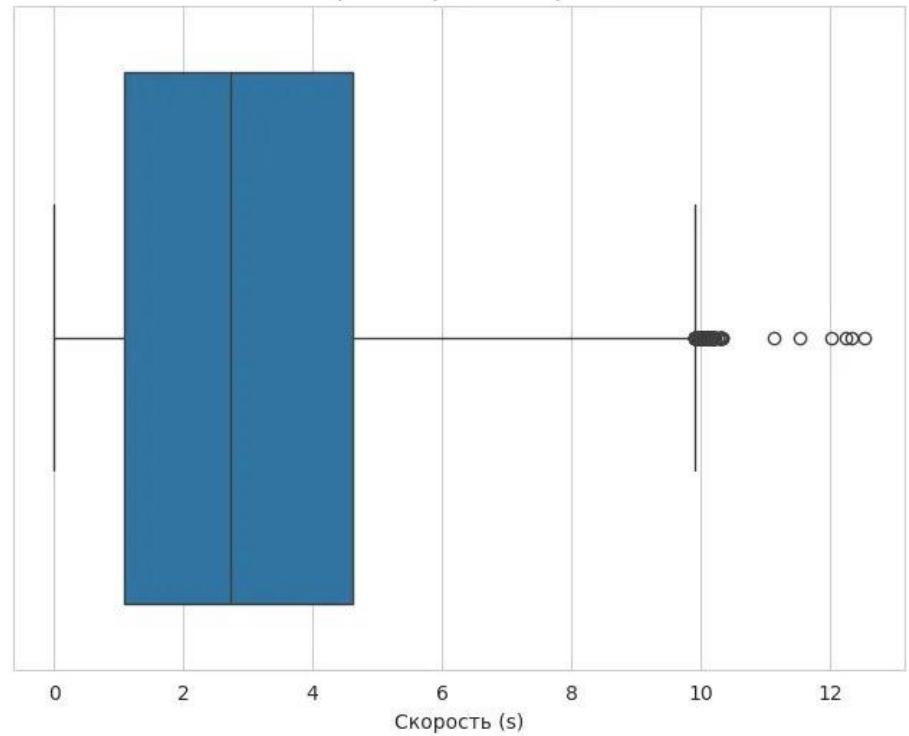
## Метод 2: Межквартильный размах (IQR)

```
python Q1, 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)]
```

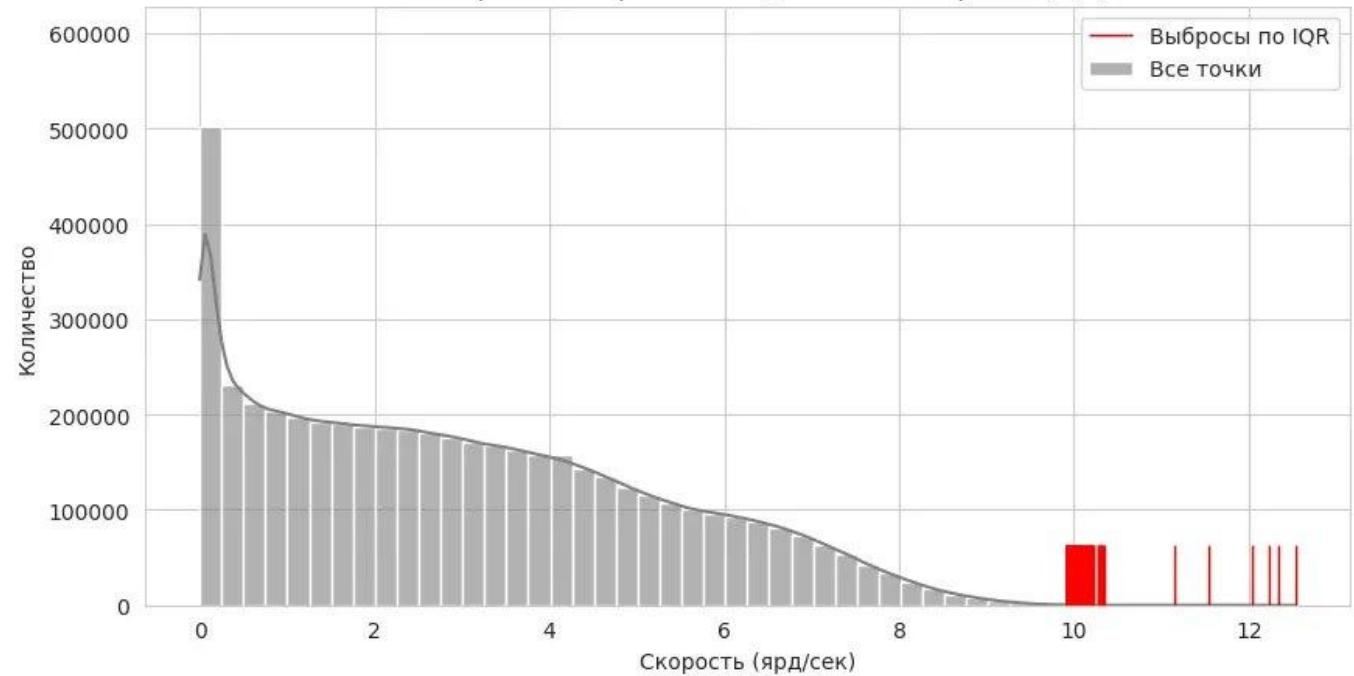
Найдено: **166 выбросов**

Более консервативный метод

Boxplot скоростей игроков



Гистограмма скорости с выделением выбросов (IQR)



# Анализ выбросов по скорости

## Физическая интерпретация:

Скорость > 10 ярдов/сек  $\approx 33$  км/ч — это спринт мирового класса

Усэйн Болт:  $\sim 12.4$  м/с  $\approx 13.5$  ярдов/сек (абсолютный максимум)

NFL игроки: реалистичный максимум  $\sim 10\text{-}11$  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)	$\sim 1,200$	0.025%
x_input	$\sim 200$	0.004%
y_input	$\sim 150$	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

**Наше 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:**

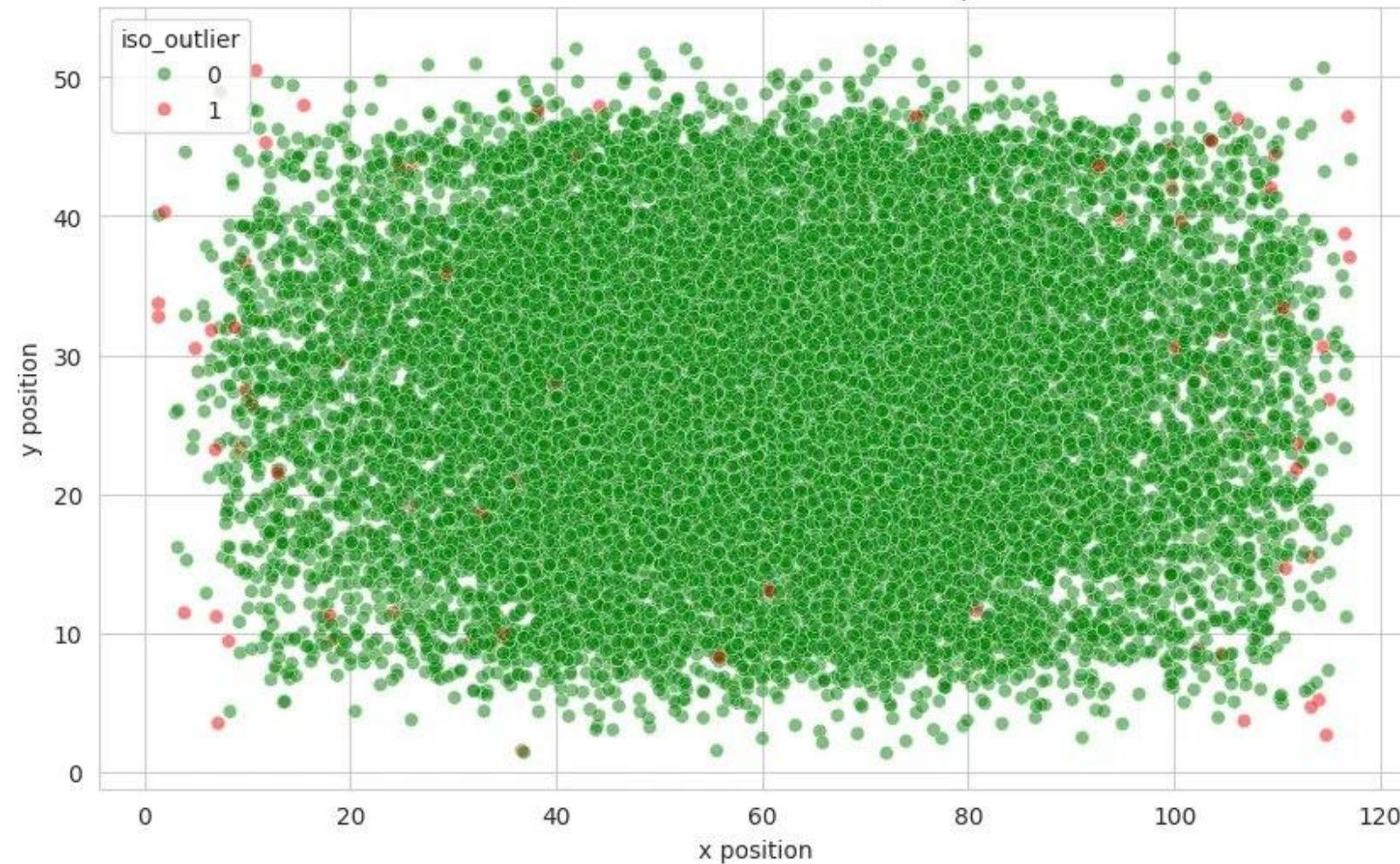
s, a, x\_input, y\_input, o, dir

**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



# 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:**

```
python df['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):**

```
python df['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!)

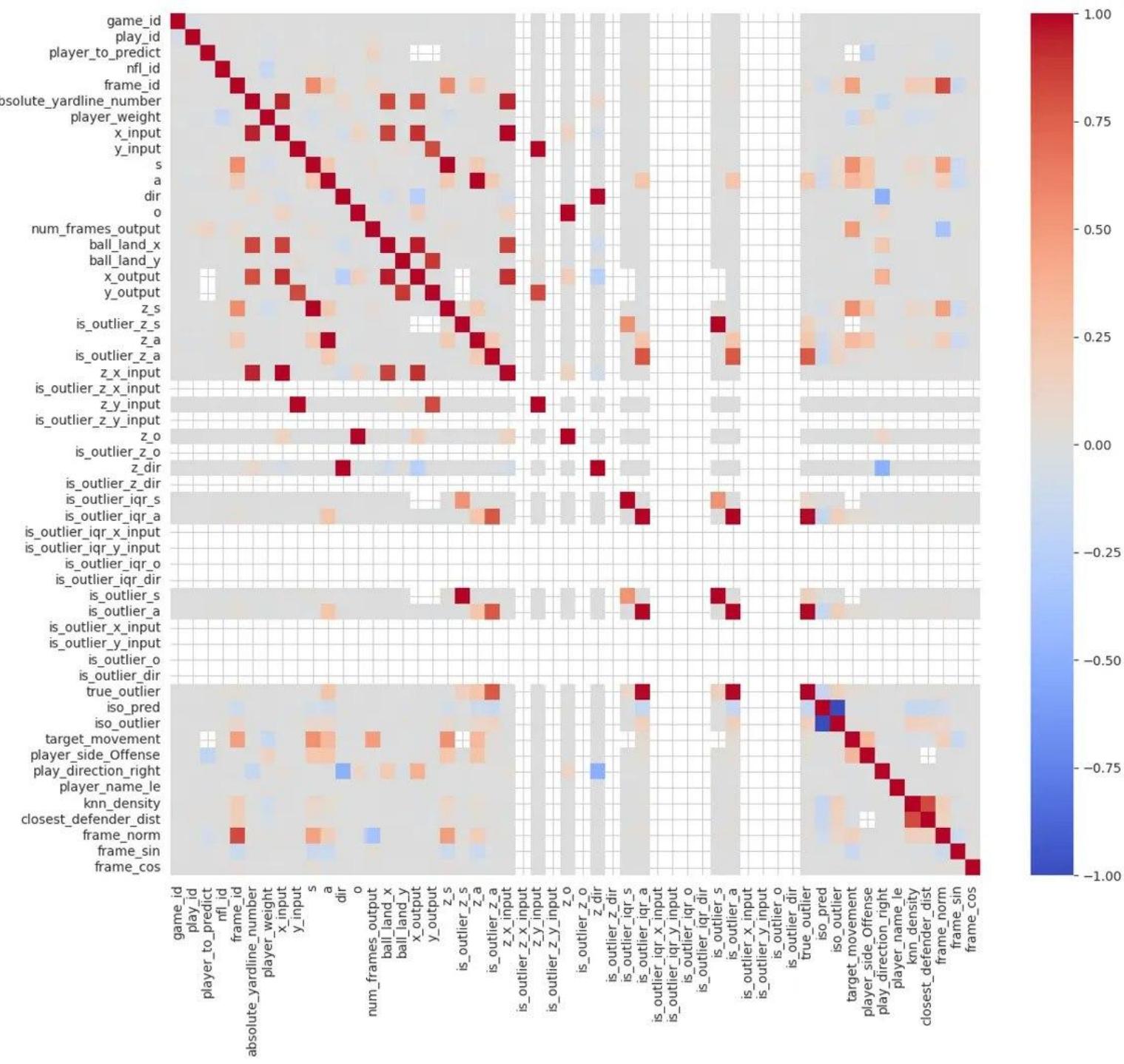
iso\_outlier имеет свою уникальную информацию

**Новые признаки:**

knn\_density коррелирует с позицией

frame\_sin/cos независимы от координат

**Вывод:** Новые признаки добавляют информацию, не дублируя старые



# Распределения координат

## Координата X (длина поля):

Распределение близко к нормальному

Пик в центре поля (40-80 ярдов)

Мало данных в эндзонах

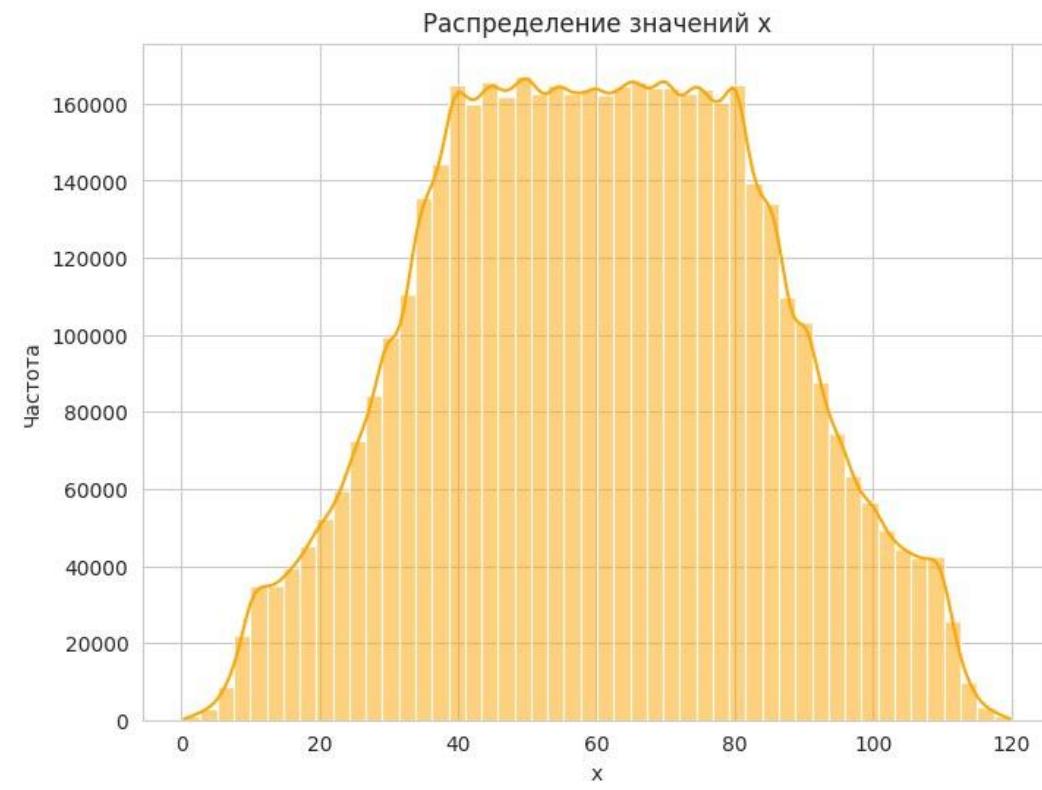
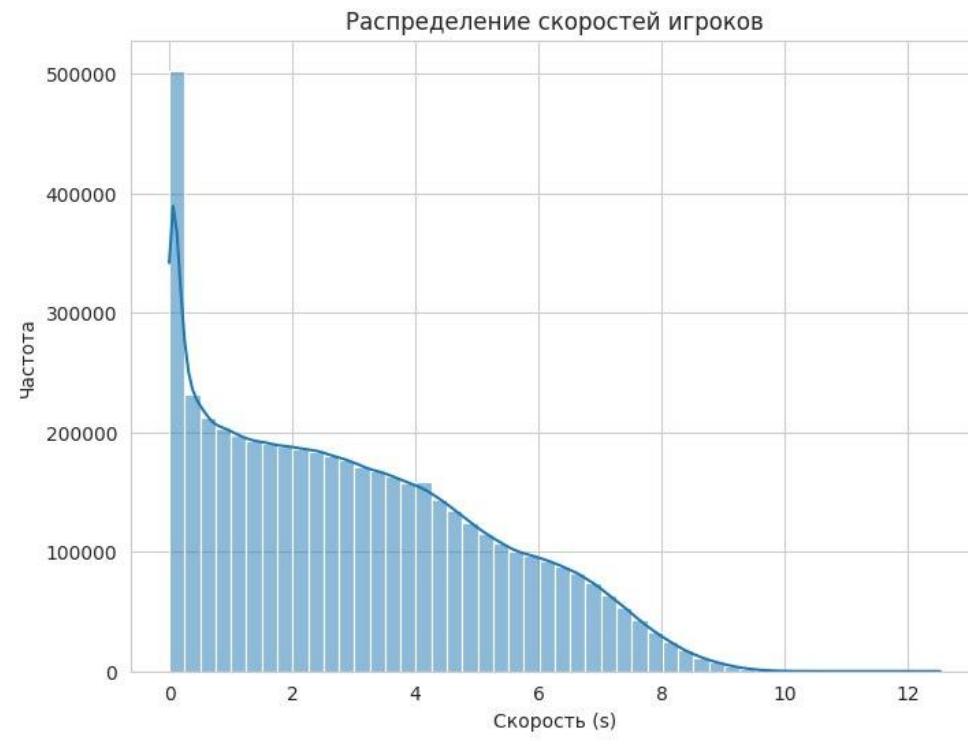
## Координата Y (ширина поля):

### **Бимодальное распределение!**

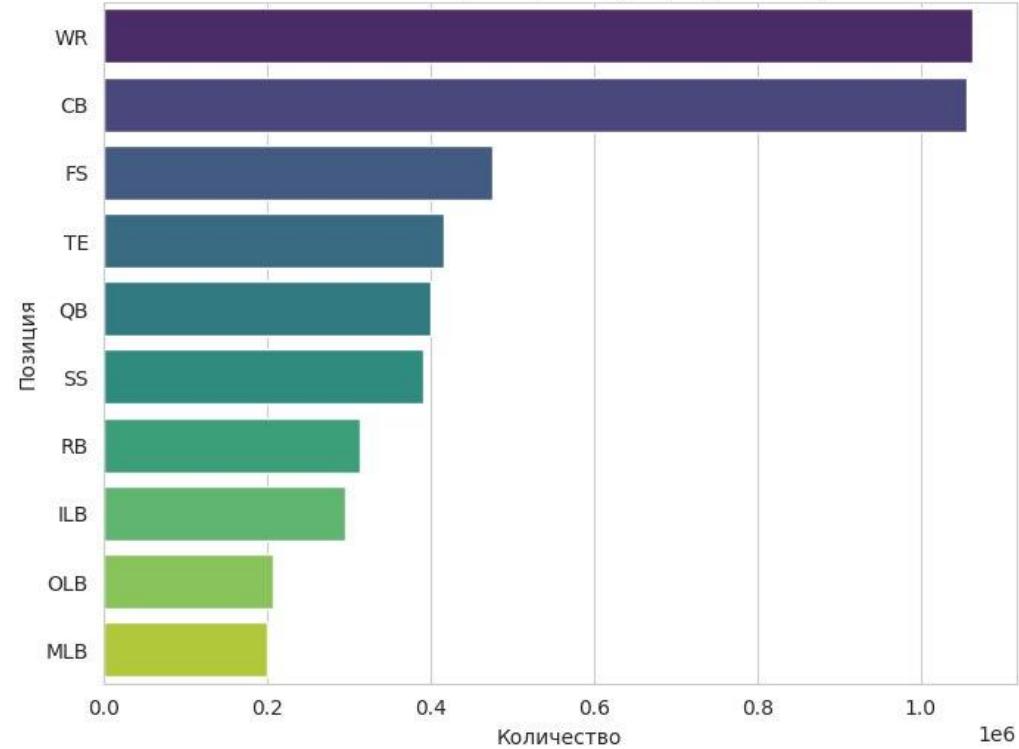
Два пика: ~20 и ~35 ярдов

Соответствует типичным построениям (hash marks)

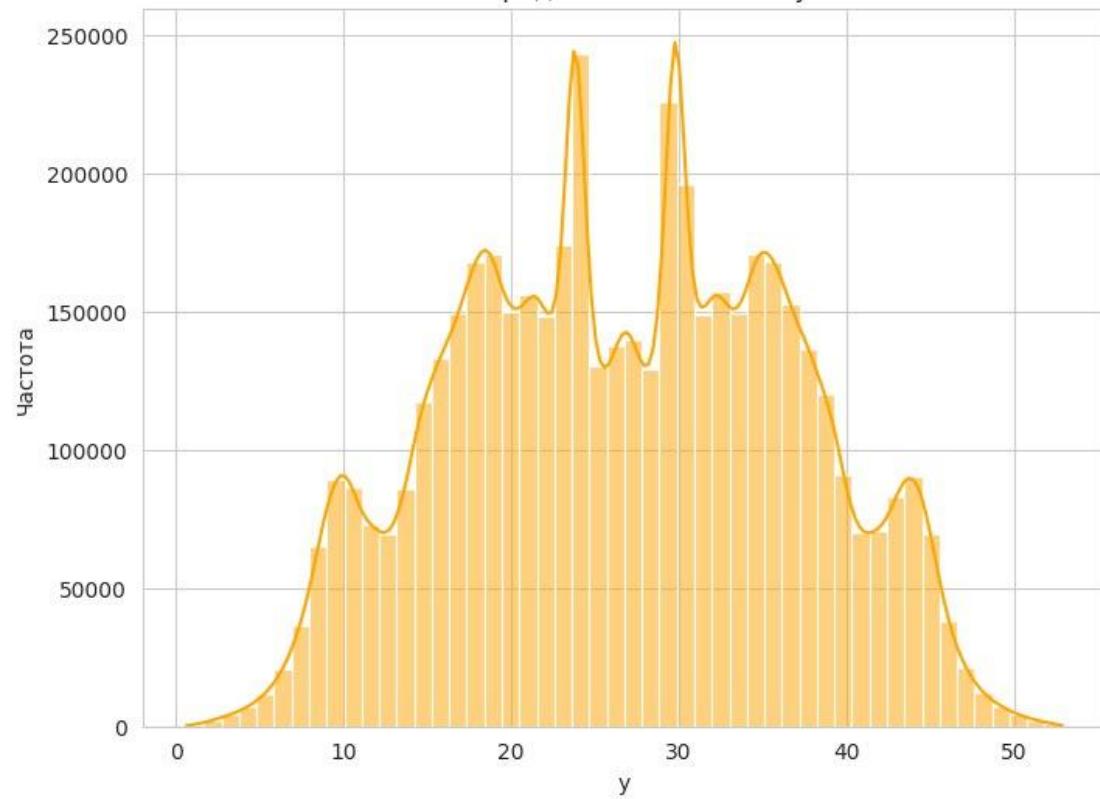
**Инсайт:** Y-координата имеет сложную структуру — модель должна это учитывать



Топ-10 позиций игроков (player\_position)



Распределение значений у



## 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	<b>18</b>	is_outlier_*, iso_outlier
KNN	<b>2</b>	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

### Новые 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 (минимальный вклад)  
LightGBM: 0.48  
CatBoost: 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

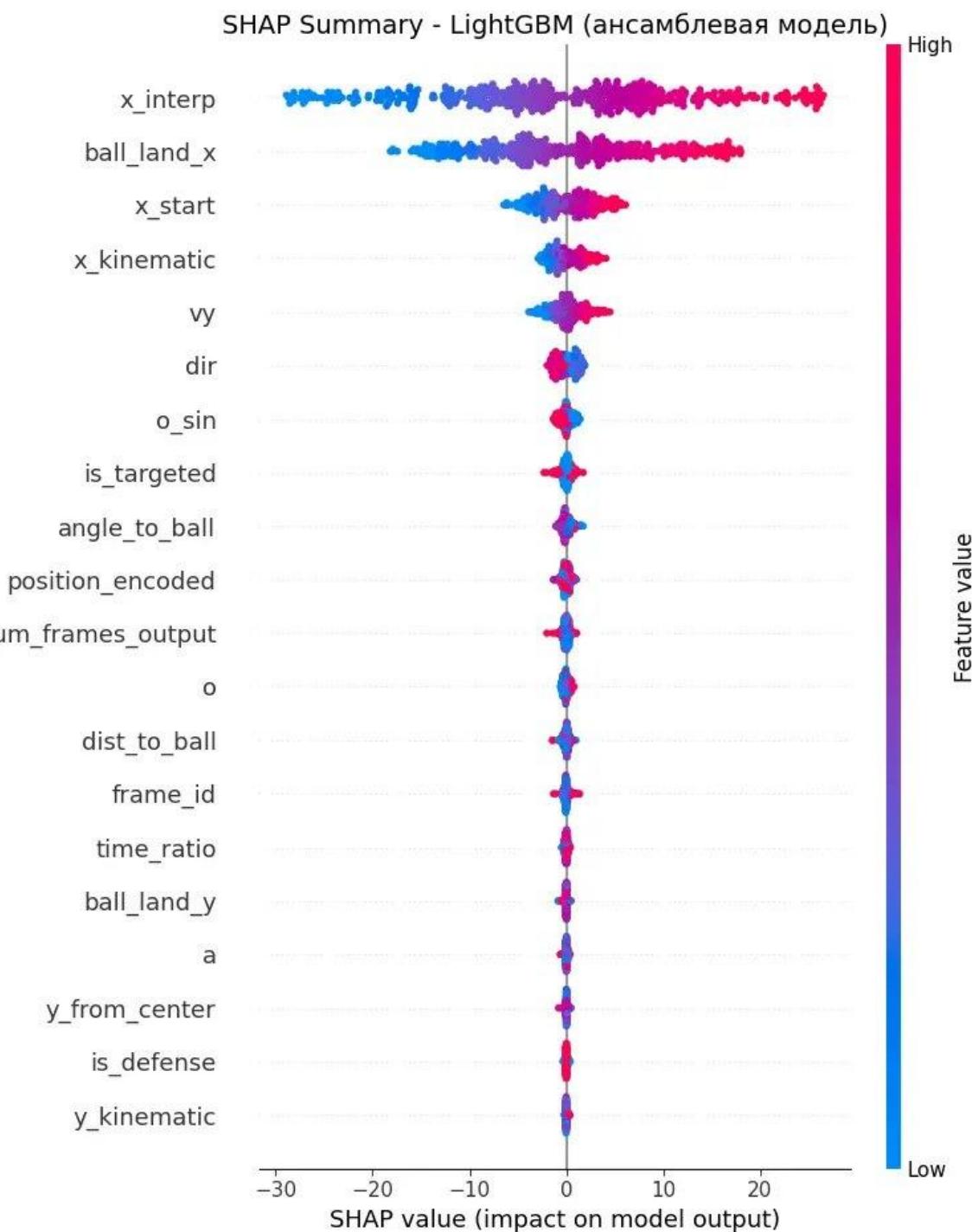
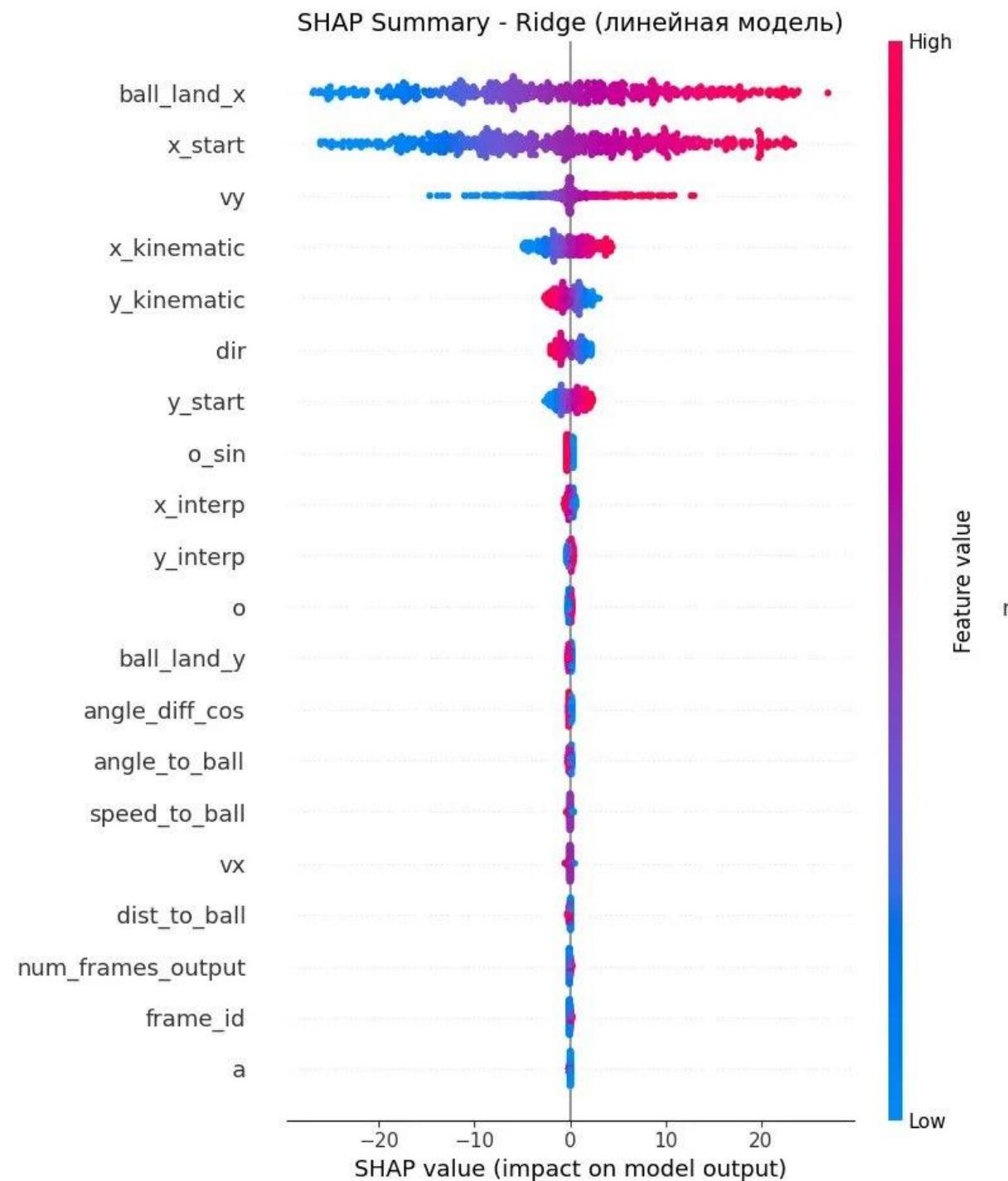
## LightGBM (ансамблевая модель):

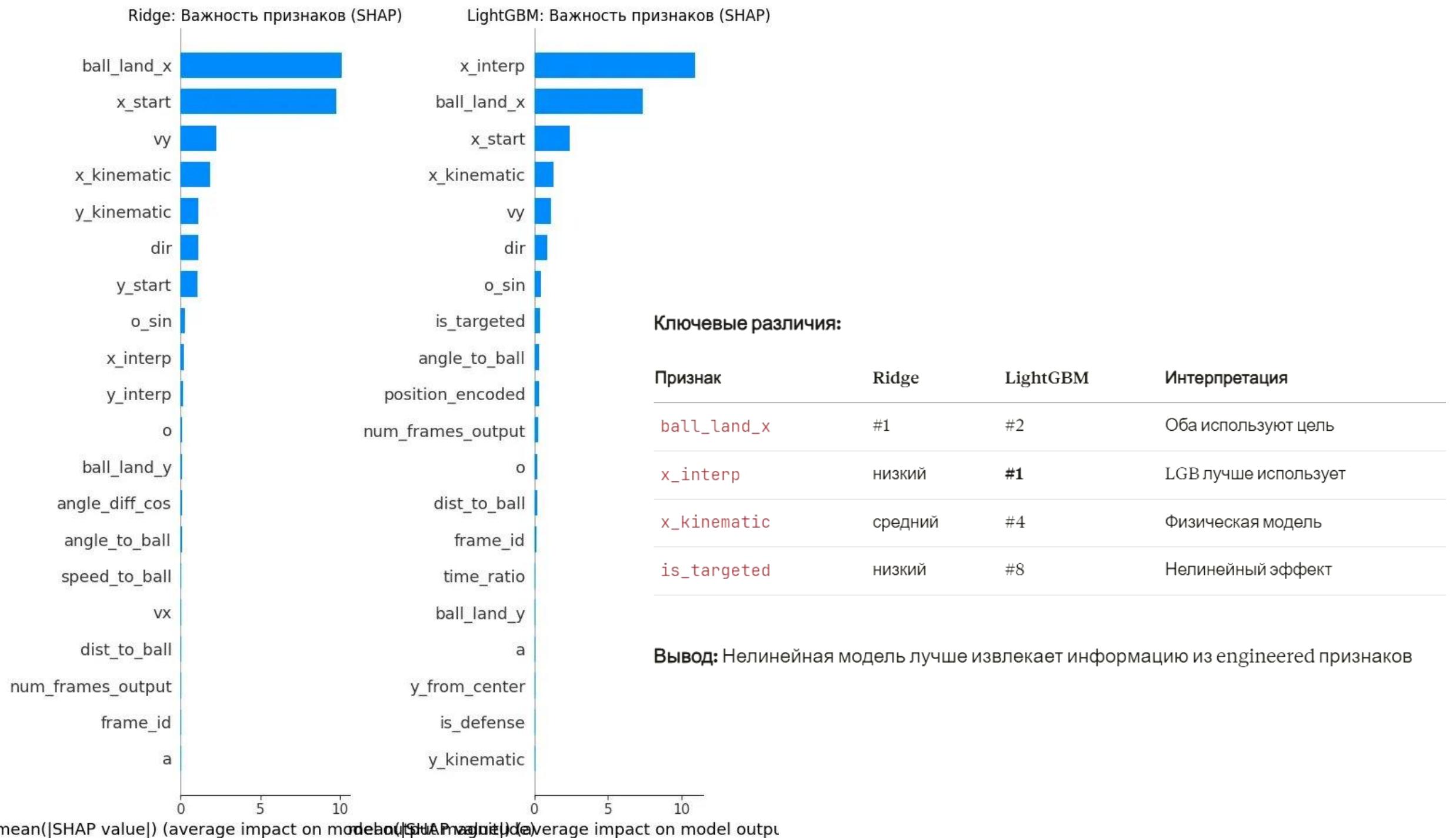
x\_interp — интерполированная позиция (наш признак!)

ball\_land\_x — место приземления

x\_start — начальная позиция

**Инсайт:** LightGBM научился использовать наш engineered признак x\_interp лучше, чем сырье данные!





# 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:  $x_{\text{start}}$  доминирует (линейная зависимость)

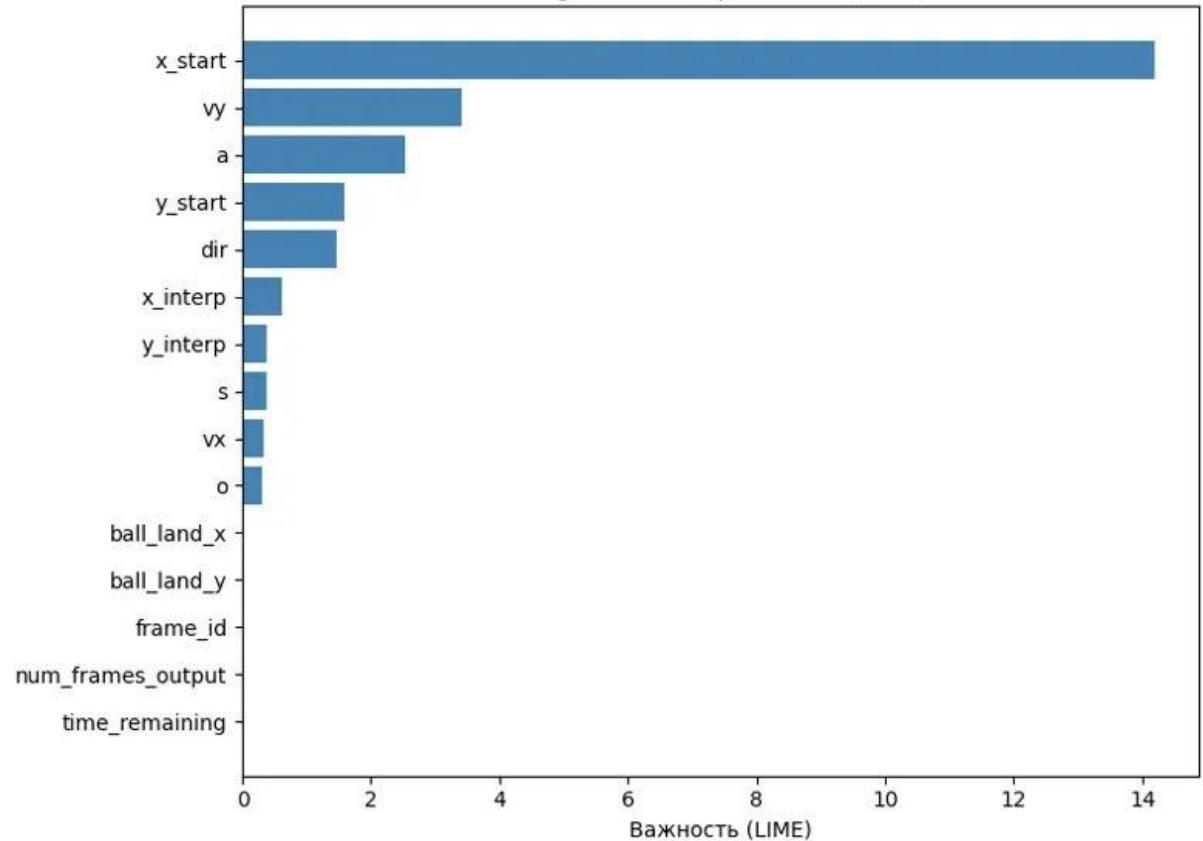
LightGBM:  $x_{\text{interp}}$  важнее (нелинейные паттерны)

## Корреляция рангов (Spearman):

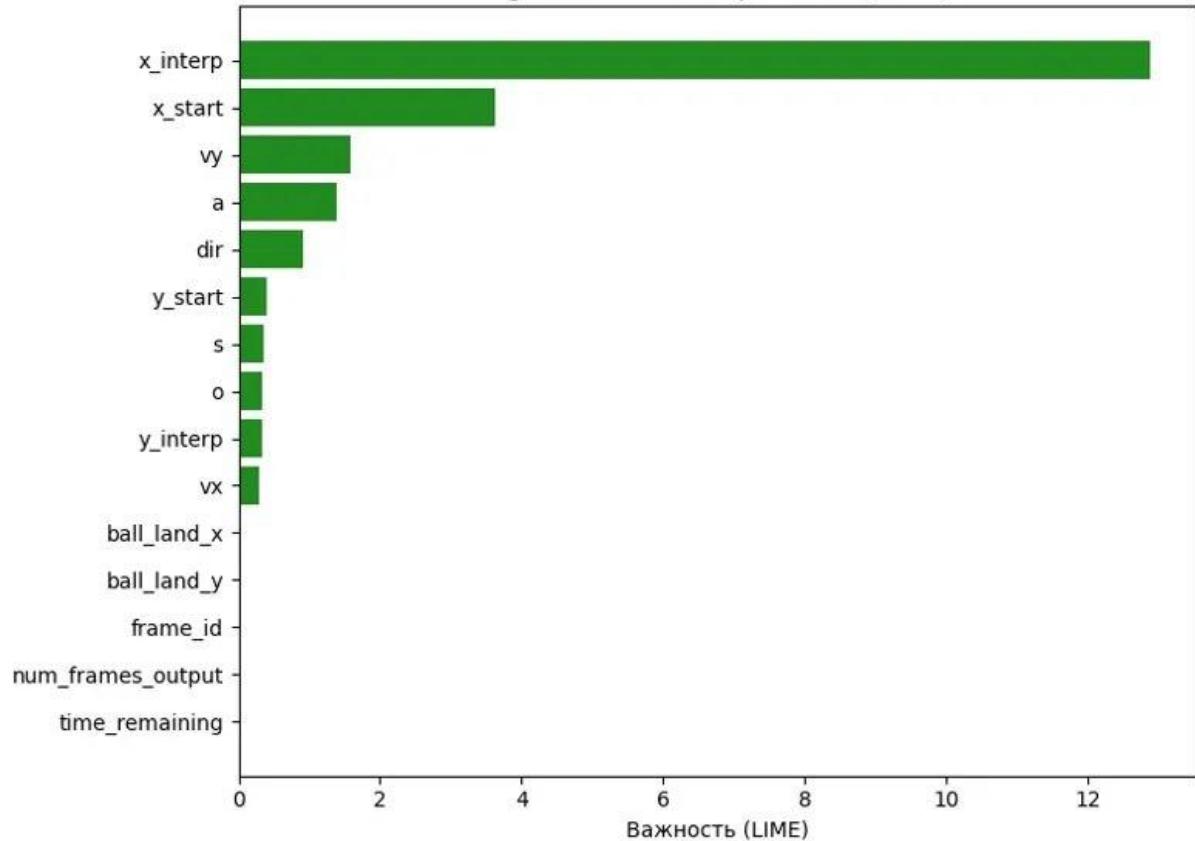
SHAP Ridge vs LightGBM:  $r = 0.65$

LIME Ridge vs LightGBM:  $r = 0.58$

Ridge: Топ-15 признаков (LIME)



LightGBM: Топ-15 признаков (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

LightGBM:

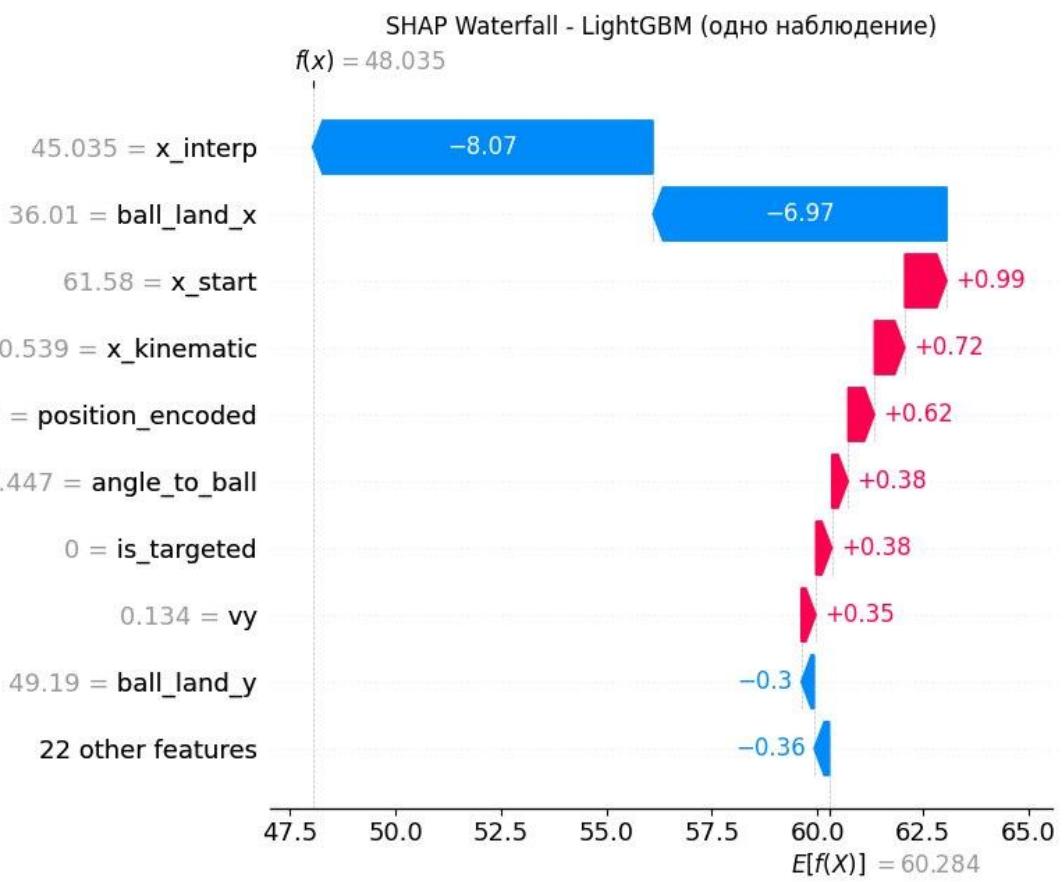
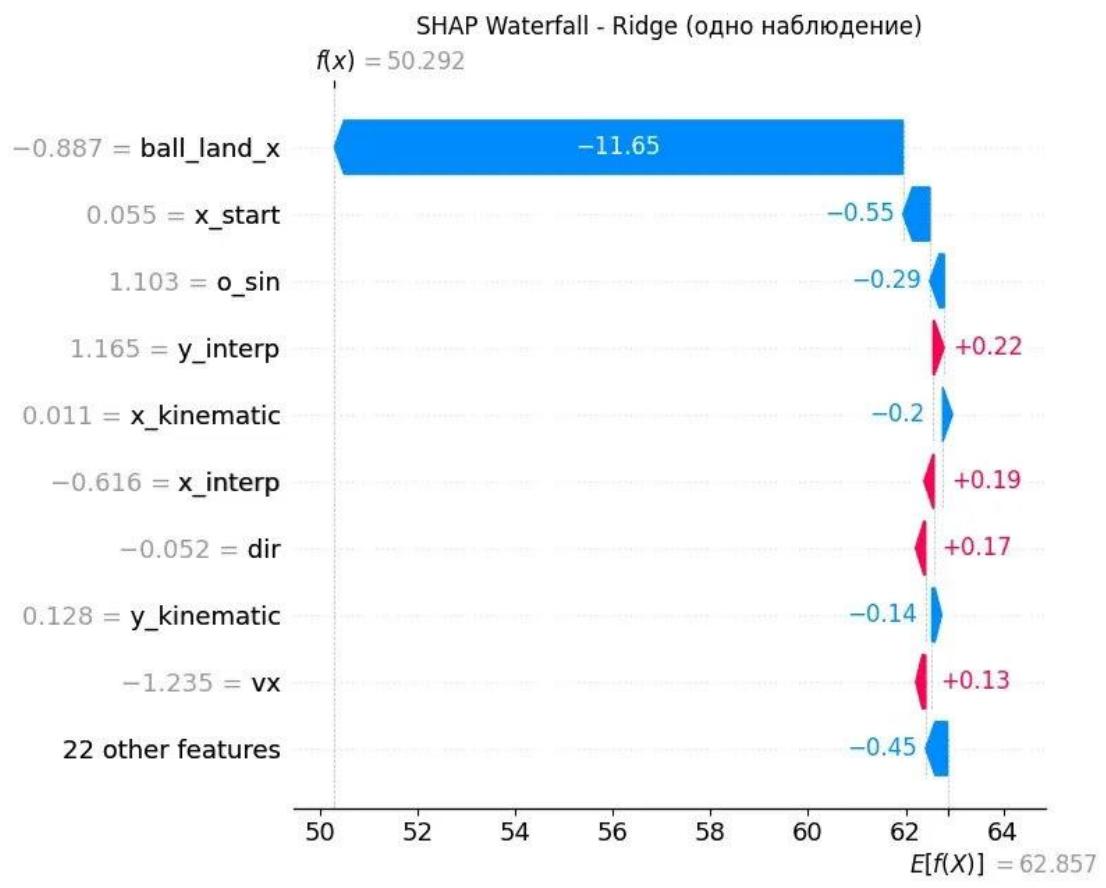
$E[f(X)] = 60.28$

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-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	<b>8.48</b>	-19.7	19.8
x_start	-0.02	<b>2.71</b>	-7.1	6.7
dir	0.01	<b>0.99</b>	-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:**

```
python combined_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

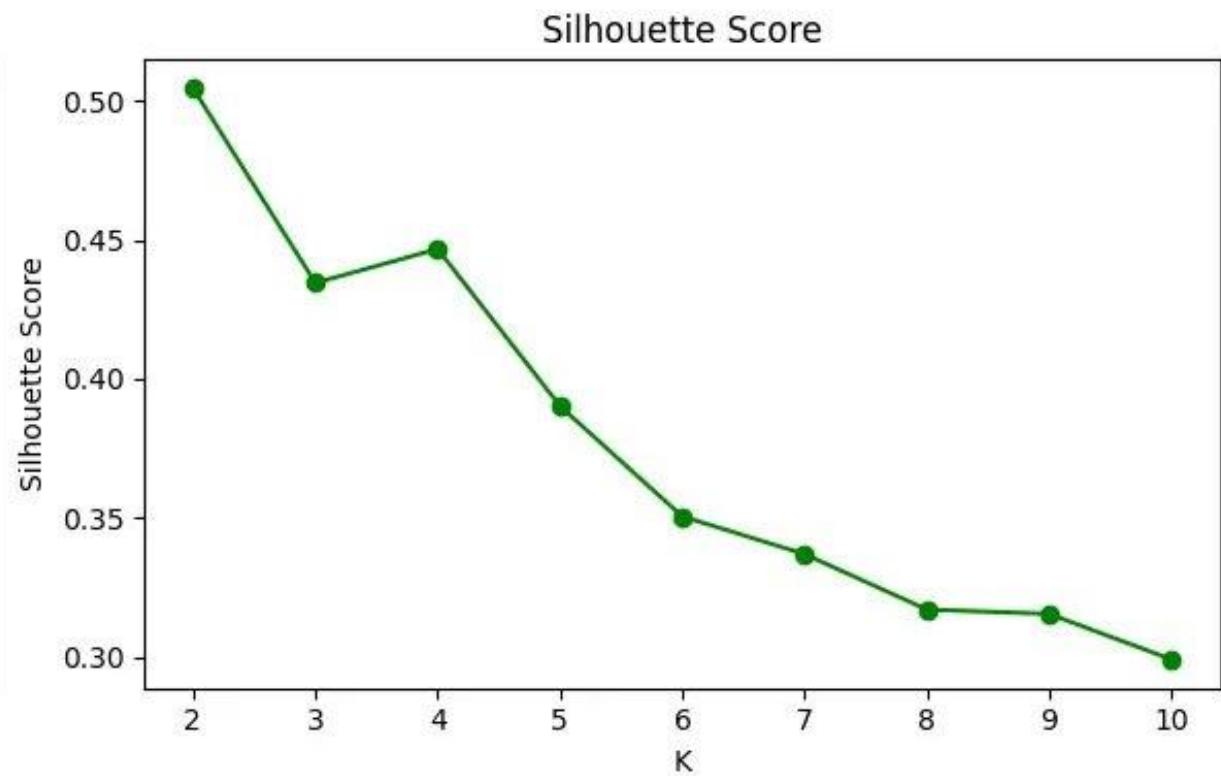
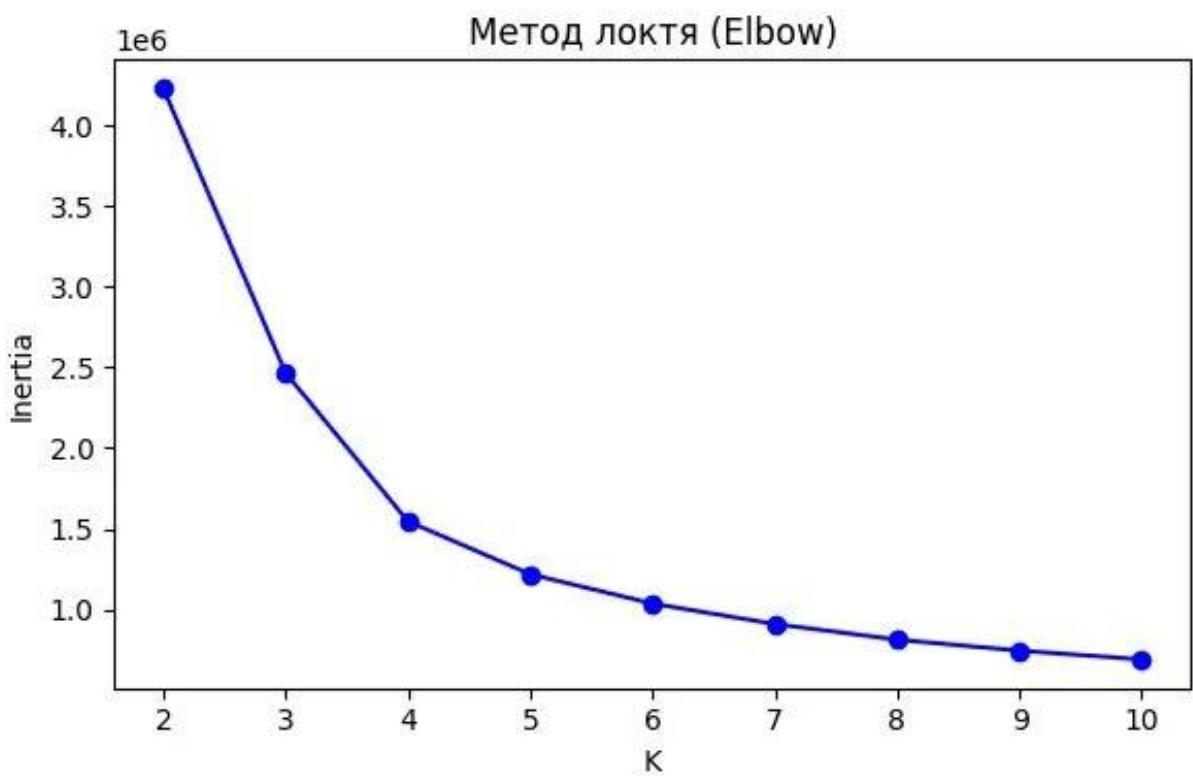
**Выбрано: K=4**

**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



# Интерпретация кластеров

**Характеристики кластеров (K-Means):**

**Инсайт: Кластеры разделяются по позиции на поле  
(x\_start)!**

**Кластер 0: правая часть поля (эндзона)**

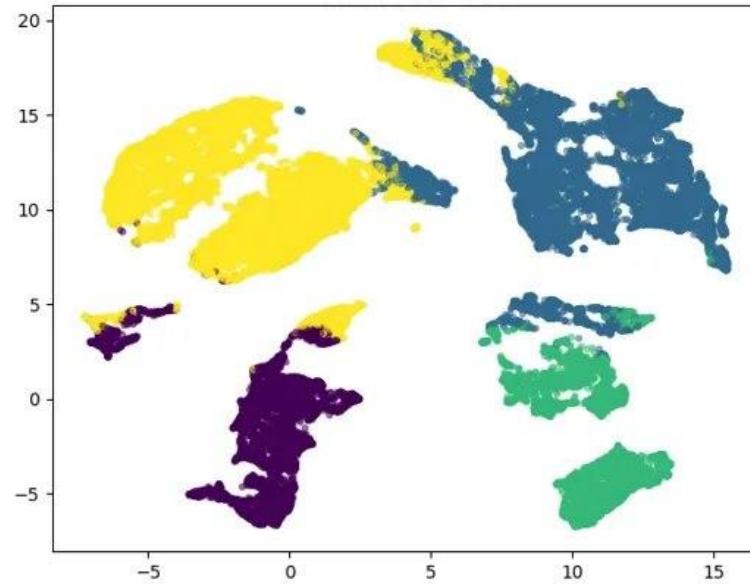
**Кластер 2: левая часть поля**

**Кластеры 1, 3: центр**

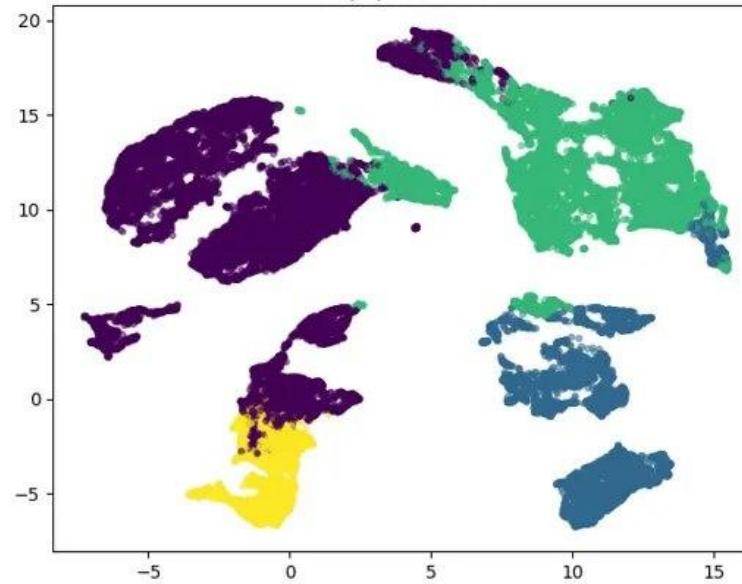
**Характеристики кластеров (K-Means):**

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

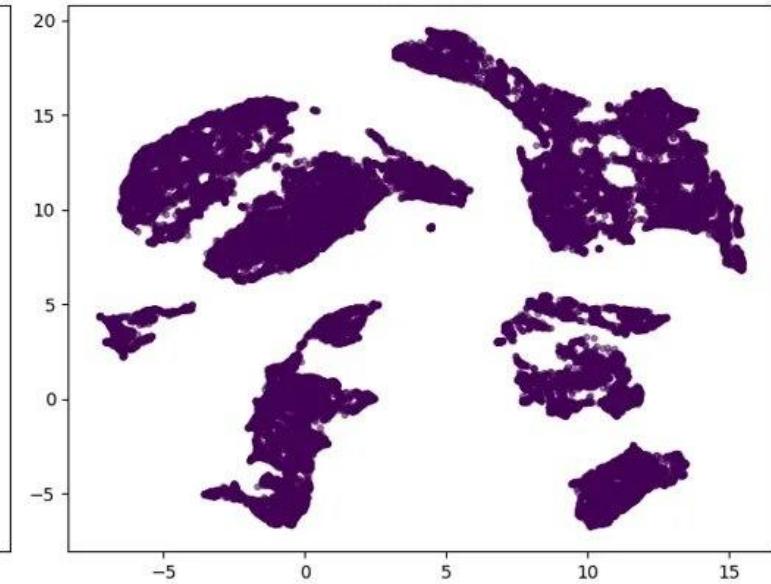
K-Means (K=4)



Иерархическая



DBSCAN



# 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:**

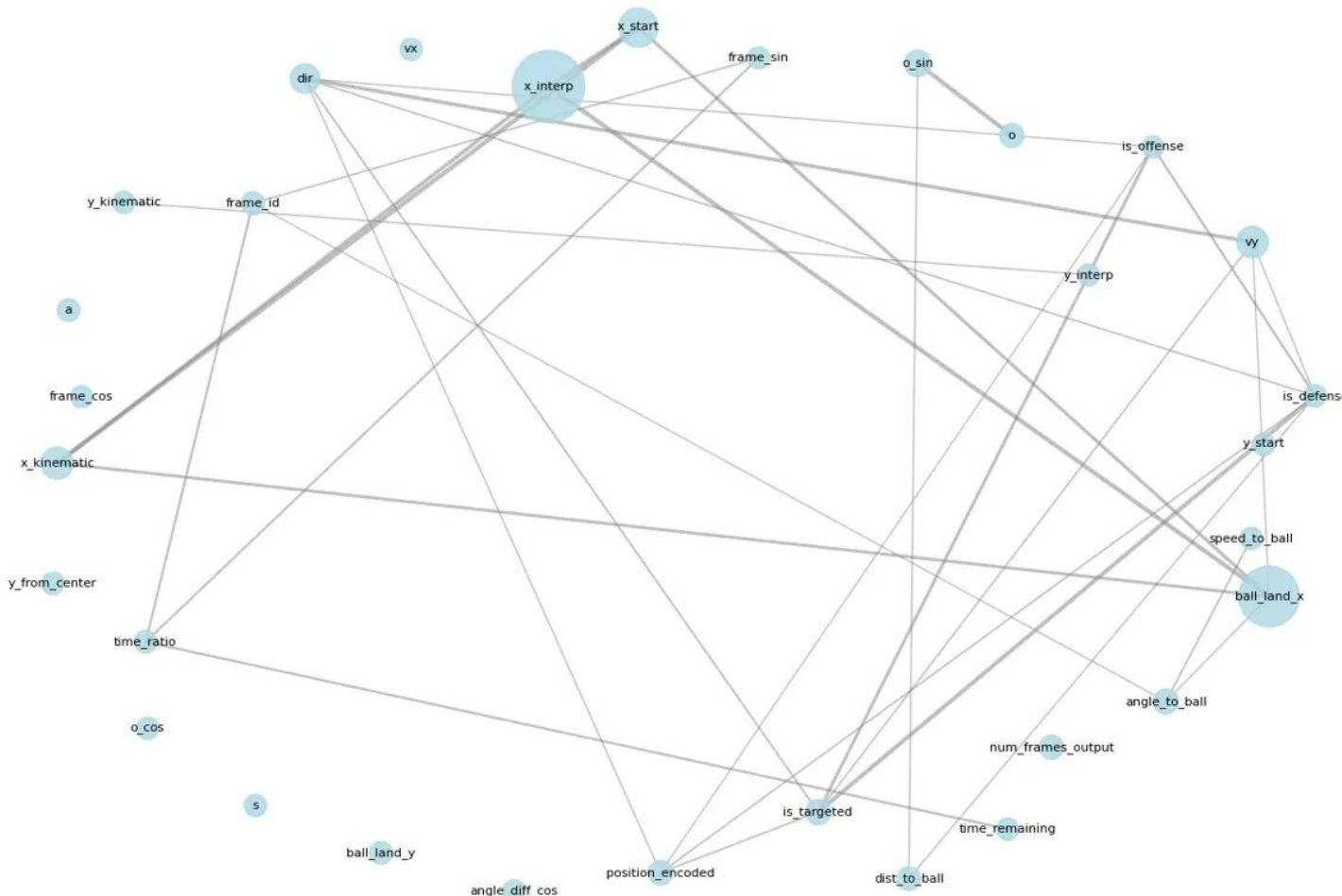
x\_interp — центральный хаб (связан с x\_start, dir, ball\_land\_x)

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



# 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	<b>3.0853</b>	±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	<b>2.87</b>	<b>-26.2%</b>

# 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%
<b>+ Optuna</b>	<b>2.87</b>	<b>-18.5%</b>

# 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**

$$w_i = (1/\text{score}_i) / \sum(1/\text{score}_j)$$

# Чем лучше модель → больше вес

**Финальное предсказание:**

$$\text{pred} = w_{\text{gru}} * \text{pred}_{\text{gru}} + w_{\text{gnn}} * \text{pred}_{\text{gnn}} + w_{\text{gru2}} * \text{pred}_{\text{gru2}}$$

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
<b>Geometric endpoint</b>	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	<b>167</b>
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)	<b>0.534</b> (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 Kaggle):**

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	<b>2.87</b>	<b>0.534</b>	5.4×
Улучшение vs baseline	-26%	—	—

500

team CU



0.534

2

17d