מדריך שיפור המודל 🚀

🎯 מטרה: שיפור הדיוק מ-68.7% ל-75%+

📊 שלב 1: הרחבת מסד הנתונים

נתונים נוספים לאיסוף:

```python
# חודשים נוספים לאיסוף
target_months = [
    (2024, 6),   # יוני 2024
    (2024, 7),   # יולי 2024
    (2024, 8),   # אוגוסט 2024 (קיים)
    (2024, 9),   # ספטמבר 2024
    (2024, 10)   # אוקטובר 2024
]

# יעד חדש: 15,000-20,000 משחקים
```

תהליך הרחבת הנתונים:

שלב 1.1: ניקוי מסד נתונים קיים

```sql
-- גיבוי הנתונים הקיימים
CREATE TABLE players_backup AS SELECT * FROM players;
CREATE TABLE games_backup AS SELECT * FROM games;

-- ניקוי לאיסוף מחדש
TRUNCATE games RESTART IDENTITY CASCADE;
TRUNCATE players RESTART IDENTITY CASCADE;
```

שלב 1.2: עדכון GameCollector

```python
```

```
# ב-game_collector.py - הוסף לולאה על חודשים
def collect_games_multiple_months(self, player_username, months_list):
    all_games = []
    for year, month in months_list:
        monthly_games = self.collect_games_for_player(player_username, year, month)
        all_games.extend(monthly_games)
    return all_games
```

שלב 1.3: איסוף הדרגתי

- התחל עם 100 שחקנים × 5 חודשים = ~5,000 משחקים נוספים
- המשך עם snowball לשחקנים נוספים
- יעד: 400+ שחקנים, 15,000+ משחקים

🔧 שלב 2: Feature Engineering מתקדם

Features חדשים לפיתוח:

2.1 Time-Based Features

```sql
sql

-- מגמת ביצועים (האם השחקן משתפר?)
CREATE VIEW player_rating_trend AS
SELECT
    player_id,
    CASE
        WHEN recent_avg > overall_avg THEN 'improving'
        WHEN recent_avg < overall_avg THEN 'declining'
        ELSE 'stable'
    END as rating_trend
FROM (
    SELECT
        player_id,
        AVG(CASE WHEN game_date >= '2024-09-01' THEN rating END) as recent_avg,
        AVG(rating) as overall_avg
    FROM player_games_view
    GROUP BY player_id
) trends;
```

2.2 Opening-Based Features

```sql
sql
```

```
-- ביצועים לפי פתיחה
CREATE VIEW opening_performance AS
SELECT
    opening_name,
    player_id,
    COUNT(*) as games_played,
    AVG(CASE WHEN result = 'white' THEN 1.0 ELSE 0.0 END) as opening_win_rate
FROM games
WHERE opening_name IS NOT NULL
GROUP BY opening_name, player_id
HAVING COUNT(*) >= 3;
```

2.3 Recent Form Features

```sql
-- ביצועים ב-10 המשחקים האחרונים
CREATE VIEW recent_form AS
SELECT
    player_id,
    AVG(CASE WHEN result = expected_winner THEN 1.0 ELSE 0.0 END) as recent_form_score
FROM (
    SELECT
        player_id,
        result,
        CASE WHEN rating > opponent_rating THEN color ELSE opponent_color END as expected_winner,
        ROW_NUMBER() OVER (PARTITION BY player_id ORDER BY game_date DESC) as game_rank
    FROM player_games_extended_view
) recent
WHERE game_rank <= 10
GROUP BY player_id;
```

2.4 Head-to-Head Features

```sql
```

```sql
-- היסטוריית מפגשים בין שחקנים
CREATE VIEW head_to_head AS
SELECT
    white_player_id,
    black_player_id,
    COUNT(*) as total_meetings,
    AVG(CASE WHEN result = 'white' THEN 1.0 ELSE 0.0 END) as white_h2h_score
FROM games
GROUP BY white_player_id, black_player_id
HAVING COUNT(*) >= 2;
```

2.5 Advanced Statistical Features

```sql
-- יציבות ביצועים (סטיית תקן של דירוגים)
CREATE VIEW player_consistency AS
SELECT
    player_id,
    STDDEV(rating) as rating_volatility,
    CASE
        WHEN STDDEV(rating) < 50 THEN 'consistent'
        WHEN STDDEV(rating) > 100 THEN 'volatile'
        ELSE 'moderate'
    END as consistency_level
FROM player_games_view
GROUP BY player_id;
```

🎛️ שלב 3: עדכון Feature Extractor

הרחבת game_features view:

```sql
```

```sql
CREATE OR REPLACE VIEW game_features_advanced AS
SELECT
    g.*,
    -- Features קיימים
    (g.white_rating - g.black_rating) as rating_diff,
    ws.white_win_rate,
    bs.black_win_rate,

    -- Features חדשים
    wt.rating_trend as white_trend,
    bt.rating_trend as black_trend,
    wf.recent_form_score as white_recent_form,
    bf.recent_form_score as black_recent_form,
    h2h.white_h2h_score,
    wc.rating_volatility as white_volatility,
    bc.rating_volatility as black_volatility,

    -- Time-based features
    EXTRACT(MONTH FROM g.game_date) as game_month,
    EXTRACT(DOW FROM g.game_date) as day_of_week

FROM games g
LEFT JOIN player_rating_trend wt ON g.white_player_id = wt.player_id
LEFT JOIN player_rating_trend bt ON g.black_player_id = bt.player_id
LEFT JOIN recent_form wf ON g.white_player_id = wf.player_id
LEFT JOIN recent_form bf ON g.black_player_id = bf.player_id
LEFT JOIN head_to_head h2h ON g.white_player_id = h2h.white_player_id
                    AND g.black_player_id = h2h.black_player_id
LEFT JOIN player_consistency wc ON g.white_player_id = wc.player_id
LEFT JOIN player_consistency bc ON g.black_player_id = bc.player_id;
```

שלב 4: שיפור המודל 🤖

4.1 רשימת Features מורחבת:

```
python
```

```
advanced_features = [
    # Features בסיסיים
    'rating_diff', 'white_win_rate', 'black_win_rate',

    # Features מתקדמים
    'white_recent_form', 'black_recent_form',
    'white_h2h_score', 'white_volatility', 'black_volatility',

    # Categorical features (after encoding)
    'white_trend_encoded', 'black_trend_encoded',
    'game_month', 'day_of_week'
]
```

4.2 עדכון ModelTrainer:

python

```
def advanced_feature_engineering(self, df):
    """Add advanced feature engineering"""

    # Encode categorical variables
    df['white_trend_encoded'] = df['white_trend'].map({
        'improving': 1, 'stable': 0, 'declining': -1
    })

    # Rating difference categories
    df['rating_diff_category'] = pd.cut(
        df['rating_diff'],
        bins=[-float('inf'), -200, -50, 50, 200, float('inf')],
        labels=['black_strong', 'black_slight', 'equal', 'white_slight', 'white_strong']
    )

    # Interaction features
    df['form_diff'] = df['white_recent_form'] - df['black_recent_form']
    df['volatility_diff'] = df['white_volatility'] - df['black_volatility']

    return df
```

4.3 אנסמבל מתקדם:

python

```
def create_ensemble_model(self, X_train, y_train):
    """Create ensemble of multiple models"""
    from sklearn.ensemble import VotingClassifier, GradientBoostingClassifier

    rf = RandomForestClassifier(n_estimators=150, max_depth=15)
    gb = GradientBoostingClassifier(n_estimators=100)
    lr = LogisticRegression()

    ensemble = VotingClassifier([
        ('rf', rf), ('gb', gb), ('lr', lr)
    ], voting='soft')

    return ensemble
```

📈 צפי לשיפור

תוצאות צפויות עם השיפורים:

שיפור	דיוק צפוי	סיבה
נתונים נוספים (15K משחקים)	+2-3%	יותר דוגמאות לאימון
Recent form features	+1-2%	הקשר לביצועים נוכחיים
Head-to-head history	+1%	דפוסים בין שחקנים ספציפיים
Ensemble models	+1-2%	שילוב מודלים משלים
סה"כ צפוי	**73-76%**	שיפור משמעותי

אתגרים צפויים:

- **איסוף נתונים**: יקח 45-60 דקות
- **Feature engineering**: SQL מורכבות יותר גבוהה
- **Overfitting**: צריך validation זהיר
- **Class imbalance**: תיקו עדיין יהיה קשה לחיזוי

⚡ סדר פעולות מומלץ

1. **יום 1**: הרחבת איסוף נתונים (5 חודשים)
2. **יום 2**: פיתוח features מתקדמים
3. **יום 3**: אימון מודלים משופרים
4. **יום 4**: אנסמבל ומסקנות סופיות

זמן משוער: 6-8 שעות עבודה נוספות