🏀 NBA Machine Learning System - Quick Start Guide

Complete Guide to Building and Using the NBA Basketball Prediction System

---

📋 Table of Contents

🎯 System Overview

This NBA machine learning system predicts **10 different NBA basketball prop types**:

1. **Points** - Total points scored
2. **Rebounds** - Total rebounds (offensive + defensive)
3. **Assists** - Total assists
4. **3-Pointers Made** - Three-point shots made
5. **Steals** - Total steals
6. **Blocks** - Total blocks
7. **Turnovers** - Total turnovers
8. **Double-Double** - Probability of 10+ in two categories
9. **Field Goals Made** - Total field goals made
10. **Free Throws Made** - Total free throws made

Key Features

✅ **Ensemble ML Models**: Linear Regression + Random Forest + Gradient Boosting
✅ **NBA-Specific Features**: Minutes, usage, home/away splits, rest days
✅ **Confidence Scores**: 0-100 scale based on model agreement
✅ **Value Finder**: Compare to betting lines and calculate expected value
✅ **Production Ready**: Automated pipeline, comprehensive logging

---

⚙️ Prerequisites

Before starting, ensure you have:

- [x] PostgreSQL database running
- [x] Database schema set up
- [x] Python 3.8+ installed
- [x] All dependencies from requirements.txt

🔧 Installation

Step 1: Navigate to Backend Directory

```
cd /home/ubuntu/betting_backend
```

Step 2: Verify ML Dependencies

```
python3 -c "import sklearn, xgboost, joblib, numpy, pandas; print('✓ All ML
dependencies installed')"
```

If you see an error, install missing packages:

```
pip install scikit-learn==1.4.0 xgboost==2.0.3 joblib==1.3.2
```

Step 3: Verify Database Connection

```
python3 -c "
from database.db_manager import db_manager
result = db_manager.execute_query('SELECT COUNT(*) FROM teams WHERE sport = %s',
('NBA',))
print(f'NBA Teams in database: {result[0][\"count\"]}')
db_manager.close()
"
```

📊 Data Collection

Before training models, you need historical NBA player game stats.

Collect Historical Data

```
# Collect all available NBA data (takes 30-60 minutes)
python collect_data.py --sport nba --with-stats

# Or collect specific date range
python collect_data.py --sport nba --start-date 2024-10-01 --end-date 2024-10-19 --
with-stats
```

Verify Data Collection

```
python3 -c "
from database.db_manager import db_manager

# Check player stats
stats_query = 'SELECT COUNT(*) FROM player_game_stats pgs JOIN games g ON pgs.game_id
= g.id WHERE g.sport = %s'
result = db_manager.execute_query(stats_query, ('NBA',))
print(f'Player game stats collected: {result[0][\"count\"]}')

# Recommendation: Need at least 500+ for good model training
if result[0]['count'] < 500:
    print('⚠️  Consider collecting more data for better model accuracy')
else:
    print('✓ Sufficient data for training')

db_manager.close()
"
```

Minimum Requirements:
- At least 500 player game stats (preferably 1000+)
- Multiple games per player (at least 10+ games per player)
- Mix of home and away games

🏋️ Training Models

Once you have sufficient data, train the NBA models.

Train All NBA Models

```
# Train models for all 10 NBA prop types
python models/nba/train_models.py
```

This will:
1. Extract features from historical data
2. Train 3 models per prop type (Linear Regression, Random Forest, Gradient Boosting)
3. Evaluate performance on test set
4. Save trained models to `models/nba/saved_models/`

Expected Output:

```
Training models for points
Train size: 1200, Test size: 300

LINEAR REGRESSION:
  MAE: 4.52
  RMSE: 6.18
  R²: 0.745

RANDOM FOREST:
  MAE: 3.98
  RMSE: 5.67
  R²: 0.782

GRADIENT BOOSTING:
  MAE: 3.85
  RMSE: 5.52
  R²: 0.795

ENSEMBLE (Average):
  MAE: 3.76
  RMSE: 5.41
  R²: 0.803

✓ Models saved successfully
```

Train Specific Props Only (Testing)

```
# Train only specific prop types for testing
python models/nba/train_models.py --prop-types points rebounds assists
```

Verify Models Were Saved

```
ls -lh models/nba/saved_models/
```

You should see `.joblib` files for each prop type and model:
- `points_linear_regression.joblib`
- `points_random_forest.joblib`
- `points_gradient_boosting.joblib`
- `points_scaler.joblib`
- `points_metadata.json`
- (and more for other prop types)

🔮 Generating Predictions

Once models are trained, generate predictions for upcoming NBA games.

Predict Today's Games

```
# Generate predictions for today's NBA games
python scripts/generate_nba_predictions.py
```

Predict Specific Date

```
# Predict games on a specific date
python scripts/generate_nba_predictions.py --date 2024-10-25
```

Predict Tomorrow's Games

```
python scripts/generate_nba_predictions.py --days-ahead 1
```

Predict Specific Props Only

```
# Generate predictions only for points, rebounds, and assists
python scripts/generate_nba_predictions.py --prop-types points rebounds assists
```

Check Data Availability Without Predicting

```
python scripts/generate_nba_predictions.py --check-only
```

Retrain Models Before Predicting

```
# Retrain models with latest data, then predict
python scripts/generate_nba_predictions.py --retrain
```

Expected Output:

```
Generating NBA Predictions for 2024-10-19

✓ Models ready
✓ Found 5 game(s) for 2024-10-19
✓ Generating predictions...

Prediction Summary
Total predictions: 150

By Prop Type:
  assists: 15 predictions (avg confidence: 72.3%)
  blocks: 15 predictions (avg confidence: 65.1%)
  fg_made: 15 predictions (avg confidence: 74.2%)
  ft_made: 15 predictions (avg confidence: 71.8%)
  points: 15 predictions (avg confidence: 78.5%)
  rebounds: 15 predictions (avg confidence: 73.7%)
  steals: 15 predictions (avg confidence: 66.4%)
  three_pt_made: 15 predictions (avg confidence: 69.9%)
  turnovers: 15 predictions (avg confidence: 67.2%)

High Confidence Predictions (>75%):
  LeBron James: points = 28.5 (confidence: 82.3%)
  Stephen Curry: three_pt_made = 4.2 (confidence: 79.1%)
  Nikola Jokic: rebounds = 11.8 (confidence: 81.7%)
  ...

✓ Successfully saved 150 predictions to database
```

💎 Finding Value Bets

Use the value finder to compare predictions against betting lines.

Python API Example

Create a file `find_value.py`:

```python
from database.db_manager import db_manager
from models.nba.predict import NBAPredictor
from models.nba.value_finder import ValueFinder

# Generate predictions
predictor = NBAPredictor()
predictions = predictor.predict_today_games()

print(f"Generated {len(predictions)} predictions")

# Create value finder
value_finder = ValueFinder()

# Example: Compare to a betting line
# (In production, you'd fetch these from an odds API)
betting_lines = {
    # Player ID: {prop_type: line}
    123: {
        'points': 25.5,
        'rebounds': 7.5,
        'assists': 8.5
    }
}

# Find predictions for this player
player_predictions = [p for p in predictions if p['player_id'] == 123]

for pred in player_predictions:
    prop_type = pred['prop_type']

    if prop_type in betting_lines[123]:
        line = betting_lines[123][prop_type]

        # Evaluate OVER bet
        over_bet = value_finder.evaluate_bet(
            prediction=pred,
            betting_line=line,
            odds=-110,
            bet_direction='over'
        )

        # Evaluate UNDER bet
        under_bet = value_finder.evaluate_bet(
            prediction=pred,
            betting_line=line,
            odds=-110,
            bet_direction='under'
        )

        # Print best bet
        if over_bet['recommendation'] == 'BET':
            print(f"\n✓ VALUE FOUND: {over_bet['player_name']} {prop_type} OVER
{line}")
            print(f"  Prediction: {over_bet['predicted_value']}")
            print(f"  Edge: {over_bet['edge']:+.1f}")
            print(f"  Expected Value: {over_bet['ev_pct']:+.1f}%")
            print(f"  Confidence: {over_bet['confidence']}%")
            print(f"  Reasoning: {over_bet['reasoning']}")

db_manager.close()
```

Run it:

```
python find_value.py
```

📈 Viewing Results

View Predictions in Database

```
psql -d betting_analysis -c "
SELECT
    p.name as player,
    proj.prop_type,
    proj.projected_value,
    proj.confidence,
    g.date
FROM projections proj
JOIN players p ON proj.player_id = p.id
JOIN games g ON proj.game_id = g.id
WHERE DATE(proj.created_at) = CURRENT_DATE
AND proj.confidence >= 75
ORDER BY proj.confidence DESC
LIMIT 10;
"
```

Python Script to View Top Predictions

```python
from database.db_manager import db_manager

query = """
SELECT
    p.name as player_name,
    p.position,
    t.abbreviation as team,
    proj.prop_type,
    proj.projected_value,
    proj.confidence,
    g.date as game_date
FROM projections proj
JOIN players p ON proj.player_id = p.id
JOIN teams t ON p.team_id = t.id
JOIN games g ON proj.game_id = g.id
WHERE DATE(proj.created_at) = CURRENT_DATE
AND proj.confidence >= 70
ORDER BY proj.confidence DESC
LIMIT 20
"""

results = db_manager.execute_query(query)

print("\n🏀 Top NBA Predictions Today\n")
print(f"{'Player':<20} {'Team':<5} {'Prop':<15} {'Prediction':<12} {'Confidence'}")
print("-" * 80)

for row in results:
    print(f"{row['player_name']:<20} {row['team']:<5} {row['prop_type']:<15} "
          f"{row['projected_value']:<12.1f} {row['confidence']:.1f}%")

db_manager.close()
```

⚙️ Automation

Set Up Daily Predictions Cron Job

Run predictions automatically every morning:

```bash
# Edit crontab
crontab -e

# Add this line (runs at 8:00 AM daily)
0 8 * * * cd /home/ubuntu/betting_backend && python3 scripts/gener-
ate_nba_predictions.py >> logs/predictions_cron.log 2>&1
```

Set Up Weekly Model Retraining

Retrain models weekly with latest data:

```bash
# Add this line to crontab (runs every Monday at 2:00 AM)
0 2 * * 1 cd /home/ubuntu/betting_backend && python3 models/nba/train_models.py >> log
s/training_cron.log 2>&1
```

View Cron Logs

```
tail -f logs/predictions_cron.log
```

🔧 Troubleshooting

Issue: “No training examples created”

Cause: Insufficient data in database

Solution:

```
# Collect more historical data
python collect_data.py --sport nba --with-stats

# Verify you have enough data
python3 -c "
from database.db_manager import db_manager
result = db_manager.execute_query('SELECT COUNT(*) FROM player_game_stats')
print(f'Player game stats: {result[0][\"count\"]}')
db_manager.close()
"
```

You need at least 500+ player game stats.

Issue: “Models not found”

Cause: Models haven’t been trained yet

Solution:

```
python models/nba/train_models.py
```

Issue: Low Model Performance (High MAE, Low R^2)

Causes:
- Not enough training data
- Data quality issues
- Need hyperparameter tuning

Solutions:
1. Collect more historical data
2. Check for missing/null values
3. Tune hyperparameters in `models/nba/config.py`

Issue: “No games found for date”

Cause: No NBA games scheduled for that date

Solution:

```
# Check what dates have games
psql -d betting_analysis -c "
SELECT date, COUNT(*)
FROM games
WHERE sport = 'NBA'
AND status = 'scheduled'
GROUP BY date
ORDER BY date;
"
```

Issue: Predictions Not Saved to Database

Check:

```python
from database.db_manager import db_manager

# Verify projections table exists
result = db_manager.execute_query("SELECT COUNT(*) FROM projections")
print(f"Projections in database: {result[0]['count']}")

# Check recent projections
result = db_manager.execute_query("""
    SELECT COUNT(*)
    FROM projections
    WHERE DATE(created_at) = CURRENT_DATE
""")
print(f"Today's projections: {result[0]['count']}")

db_manager.close()
```

📊 Understanding the Output

Prediction Format

Each prediction includes:
- **predicted_value**: Model's prediction
- **confidence_score**: 0-100 (higher = more reliable)
- **prediction_low**: Lower bound of prediction interval
- **prediction_high**: Upper bound of prediction interval
- **model_predictions**: Individual model predictions

Confidence Score Breakdown

- **80-100**: Very high confidence (strong bet consideration)
- **70-79**: High confidence (good bet consideration)
- **60-69**: Moderate confidence (proceed with caution)
- **Below 60**: Low confidence (avoid betting)

Model Performance Metrics

- **MAE (Mean Absolute Error)**: Average prediction error
- Points: ~3.8 (predicts within 3.8 points on average)
- Rebounds: ~1.9

- Assists: ~1.5
- **R^2 Score**: How well model fits data (0-1)
- 0.80 = Very good
- 0.70 = Good
- 0.60 = Acceptable
- **Within 3 Points**: Percentage of predictions within 3 points of actual
- Points: ~56%
- Assists: ~64%

🎓 Advanced Usage

Custom Feature Engineering

Edit `models/nba/feature_engineering.py` to add custom features:

```
def extract_features_for_player(self, ...):
    # Add your custom feature
    features['my_custom_metric'] = self.calculate_custom_metric(stats_list)
    return features
```

Hyperparameter Tuning

Edit `models/nba/config.py`:

```
MODEL_PARAMS = {
    'random_forest': {
        'n_estimators': 200,  # Increase trees
        'max_depth': 15,      # Increase depth
        ...
    }
}
```

Then retrain:

```
python models/nba/train_models.py
```

📚 Additional Resources

- **Detailed Model Documentation**: `models/nba/README.md`
- **Database Schema**: `database/schema.sql`
- **Main Backend README**: `README.md`
- **Example Usage**: `models/nba/example_usage.py`

🚀 Quick Command Reference

```
# Data Collection
python collect_data.py --sport nba --with-stats

# Training
python models/nba/train_models.py

# Predictions
python scripts/generate_nba_predictions.py

# View Predictions
psql -d betting_analysis -c "SELECT * FROM projections_view WHERE DATE(created_at) =
CURRENT_DATE LIMIT 10;"

# Check Logs
tail -f logs/train_models.log
tail -f logs/predict.log
```

⚠️ Important Notes

1. **Data Quality**: Model accuracy depends on data quality and quantity
2. **Rookies**: New players may not have enough data for predictions
3. **Mid-Season**: Models are more accurate mid-season with more data
4. **Responsible Betting**: Always verify predictions with your own analysis
5. **Educational Purpose**: This system is for research and educational purposes

---

🎯 Next Steps

After completing this quickstart:

1. ✅ Collect historical NBA data
2. ✅ Train models for all prop types
3. ✅ Generate daily predictions
4. ✅ Compare to betting lines
5. ✅ Set up automation
6. ✅ Monitor model performance
7. ✅ Retrain weekly with new data

Happy predicting! 🏀

Remember: Sports betting involves risk. Always bet responsibly and within your means.