Backtesting Engine - Quick Start Guide

🎯 Overview

A production-ready backtesting engine has been built at `/home/ubuntu/betting_backend/backtesting/` to test betting strategies on historical data and generate actionable insights.

📁 Structure

```
backtesting/
├── README.md                    # Comprehensive documentation
├── __init__.py                  # Package initializer
├── config.py                    # All configuration (payouts, thresholds, strategies)
├── strategy_simulator.py        # Betting strategy simulator
├── performance_analyzer.py      # Performance analysis across dimensions
├── insights_generator.py        # Actionable insights generator
├── run_backtest.py             # Main CLI runner (executable)
└── api.py                      # Dashboard API functions

database/migrations/
└── add_backtest_results.sql    # Database schema migration

scripts/
└── run_weekly_backtest.py      # Weekly automation script
```

🚀 Quick Start

1. Setup Database

Run the migration to create the `backtest_results` table:

```
cd /home/ubuntu/betting_backend
psql -U betting_user -d betting_analysis -f database/migrations/
add_backtest_results.sql
```

2. Run Your First Backtest

Test a confidence-based strategy on the last 30 days:

```
python backtesting/run_backtest.py \
  --start-date 2024-09-19 \
  --end-date 2024-10-19 \
  --strategy confidence_based \
  --confidence-threshold 75 \
  --save
```

3. View Results

Results are automatically displayed in the console and saved to the database if `--save` is used.

📊 Available Strategies

1. Confidence-Based

Bets only when model confidence exceeds threshold.

```
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy confidence_based \
  --confidence-threshold 75 \
  --save
```

2. Value-Based

Bets only when expected value exceeds threshold.

```
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy value_based \
  --ev-threshold 10 \
  --save
```

3. Prop-Specific

Focuses on specific prop types.

```
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy prop_specific \
  --props points assists \
  --sport NBA \
  --save
```

4. Composite

Combines multiple filters for refined selection.

```
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy composite \
  --confidence-threshold 75 \
  --ev-threshold 5 \
  --entry-sizes 2 3 4 5 \
  --save
```

🎰 Entry Type Payouts

- **2-pick**: 3x payout
- **3-pick**: 6x payout
- **4-pick**: 10x payout
- **5-pick**: 20x payout

💡 Key Features

Strategy Simulation

- Multiple betting strategies (confidence, value, prop-specific, composite)
- Realistic bet generation from historical predictions
- Accurate outcome evaluation against actual results

Bankroll Management

- **Flat Betting**: Fixed bet size
- **Percentage**: Bet % of bankroll
- **Kelly Criterion**: Optimal sizing based on edge

Performance Analysis

- Win rate by entry size, prop type, sport, confidence
- ROI, Sharpe ratio, max drawdown, profit factor
- Best prop combinations for parlays
- Risk-adjusted returns

Insights Generation

- Automatically identifies strengths/weaknesses
- Provides specific recommendations
- Highlights optimal strategies
- Categorized by priority (high/medium/low)

🔧 Common Commands

Compare Sports

```
# NBA only
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy confidence_based --sport NBA --save

# NFL only
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy confidence_based --sport NFL --save
```

Test Different Entry Sizes

```
# Focus on 2-pick and 3-pick
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy confidence_based \
  --entry-sizes 2 3 \
  --save
```

Custom Bankroll Settings

```
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy confidence_based \
  --bankroll 2000 \
  --bet-size 100 \
  --bankroll-strategy flat \
  --save
```

Save to JSON File

```
python backtesting/run_backtest.py \
  --start-date 2024-09-01 --end-date 2024-10-19 \
  --strategy composite \
  --confidence-threshold 80 \
  --output backtest_results.json
```

🤖 Automation

Run weekly backtests automatically:

```
python scripts/run_weekly_backtest.py
```

This script:
- Tests multiple strategies on last 30 days
- Tests NBA and NFL separately
- Saves all results to database
- Generates summary report
- Identifies best performers

Schedule with Cron

```
# Run every Monday at 6 AM
0 6 * * 1 cd /home/ubuntu/betting_backend && python scripts/run_weekly_backtest.py >>
logs/weekly_backtest.log 2>&1
```

📈 Dashboard Integration

Use the API to integrate with your dashboard:

```python
from backtesting.api import (
    get_strategy_performance,
    get_entry_size_analysis,
    get_prop_type_performance,
    get_sport_comparison,
    get_key_insights,
    get_historical_chart_data,
    get_backtest_summary
)

# Get latest strategy performance
strategies = get_strategy_performance(sport='NBA', limit=10)

# Get key insights
insights = get_key_insights(limit=5)

# Get sport comparison
comparison = get_sport_comparison()

# Get chart data for visualization
chart_data = get_historical_chart_data(chart_type='cumulative_pl')
```

📊 Understanding Results

Performance Metrics

Win Rate
- Good: 55%+
- Excellent: 60%+

ROI (Return on Investment)
- Good: 5%+
- Excellent: 15%+

Sharpe Ratio (risk-adjusted returns)
- Good: 1.0+
- Excellent: 2.0+

Max Drawdown (largest decline)
- Good: <20%
- Concerning: >30%

Profit Factor (gross profit / gross loss)
- Good: 1.5+
- Excellent: 2.0+

Insights Categories

✅ **Success**: Positive findings
⚠️ **Warning**: Areas of concern
ℹ️ **Info**: Neutral observations

Priority Levels:
- **High**: Critical findings requiring action
- **Medium**: Important but not urgent
- **Low**: Nice-to-know information

🎯 Example Output

```
================================================================================
BACKTEST RESULTS: CONFIDENCE_BASED
================================================================================

📊 Performance Summary:
  Total Bets: 150
  Wins: 85 | Losses: 65
  Win Rate: 56.67%
  Total Profit: $1,245.50
  ROI: 16.61%
  Max Drawdown: 12.34%
  Sharpe Ratio: 1.85
  Bankroll: $1,000.00 → $2,245.50

💡 Key Insights:
  1. ✅ Strong Win Rate
     Your strategy achieved a 56.67% win rate, which is excellent for sports betting.

  2. ✅ 3-Pick Entries Perform Best
     3-pick entries have the highest ROI at 22.5% with a 60% win rate.
     → Focus on 3-pick entries for optimal returns.

  3. ℹ️ NBA Outperforms NFL
     NBA props have 8.5% higher win rate than NFL (58% vs 49.5%).
     → Allocate more bankroll to NBA props for better returns.

  4. ✅ Points and Assists Props Lead
     Points, Assists have the highest win rates (68%+).
     → Prioritize Points and Assists props in your entries.

  5. ✅ Strategy Ready for Live Betting
     Strong performance metrics suggest this strategy is viable for real betting.
     → Start with small stakes and gradually scale up as you validate results.
```

🛠️ Troubleshooting

No predictions found

- Verify date range has completed games
- Check predictions exist in database
- Run prediction generation if needed

Database connection errors

- Check database credentials in config
- Ensure PostgreSQL is running
- Verify schema is installed

Low bet count

- Lower confidence threshold
- Expand date range
- Include both sports
- Remove restrictive filters

📚 Full Documentation

For complete documentation, see:
- `backtesting/README.md` - Full technical documentation
- `backtesting/config.py` - All configuration options
- `backtesting/api.py` - Dashboard integration functions

🎓 Best Practices

1. **Start with sufficient data**: Minimum 30 days, 50+ bets
2. **Test multiple strategies**: Compare different approaches
3. **Use conservative bankroll**: Don't bet more than 5% per entry
4. **Validate before live**: Paper trade first, start small
5. **Monitor and adjust**: Run weekly backtests, track over time
6. **Avoid overfitting**: Keep strategies simple and robust

🚦 Next Steps

1. ✅ Run database migration
2. ✅ Run first backtest with default settings
3. ✅ Test different strategies and compare
4. ✅ Integrate with dashboard using API
5. ✅ Set up weekly automation
6. ✅ Monitor performance and adjust

💻 Command Reference

```
# Help
python backtesting/run_backtest.py --help

# Basic run
python backtesting/run_backtest.py \
  --start-date YYYY-MM-DD \
  --end-date YYYY-MM-DD \
  --strategy [confidence_based|value_based|prop_specific|composite] \
  --save

# Full options
python backtesting/run_backtest.py \
  --start-date YYYY-MM-DD \
  --end-date YYYY-MM-DD \
  --strategy STRATEGY \
  --sport [NBA|NFL] \
  --confidence-threshold INT \
  --ev-threshold FLOAT \
  --props PROP1 PROP2 ... \
  --entry-sizes 2 3 4 5 \
  --bankroll FLOAT \
  --bet-size FLOAT \
  --bankroll-strategy [flat|percentage|kelly] \
  --save \
  --output FILE.json \
  --verbose
```

Ready to start backtesting! 🚀

For questions or issues, check the logs at `logs/backtesting.log` or review the comprehensive documentation in `backtesting/README.md`.