Contributing to ProFootballAI

First off, thank you for considering contributing to ProFootballAI! 🎉

The following is a set of guidelines for contributing to ProFootballAI. These are mostly guidelines, not rules. Use your best judgment, and feel free to propose changes to this document in a pull request.

Table of Contents

Code of Conduct

This project and everyone participating in it is governed by our Code of Conduct. By participating, you are expected to uphold this code. Please report unacceptable behavior to conduct@profootball-ai.com.

Our Standards

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Getting Started

Prerequisites

1. Python 3.9 or higher
2. Git
3. A GitHub account
4. API-Football key (for testing)

Setting Up Your Development Environment

1. **Fork the repository** on GitHub
2. **Clone your fork** locally:

   ```bash
   git clone https://github.com/your-username/profootball-ai.git
   cd profootball-ai
   ```

3. **Add upstream remote**:

   ```bash
   git remote add upstream https://github.com/original/profootball-ai.git
   ```

4. **Create a virtual environment**:

   ```bash
   python -m venv venv
   source venv/bin/activate  # On Windows: venv\Scripts\activate
   ```

5. **Install dependencies**:

   ```bash
   pip install -r requirements-dev.txt
   pre-commit install
   ```

6. **Set up environment**:

   ```bash
   cp .env.example .env
   # Edit .env with your settings
   ```

7. **Initialize database**:

   ```bash
   python scripts/init_database.py --populate
   ```

8. **Run tests** to ensure everything is working:

   ```bash
   pytest
   ```

How Can I Contribute?

Reporting Bugs

Before creating bug reports, please check existing issues as you might find out that you don't need to create one. When you are creating a bug report, please include as many details as possible:

- **Use a clear and descriptive title**
- **Describe the exact steps to reproduce the problem**
- **Provide specific examples**
- **Describe the behavior you observed**
- **Explain which behavior you expected to see**
- **Include screenshots if possible**
- **Include your environment details** (OS, Python version, etc.)

Suggesting Enhancements

Enhancement suggestions are tracked as GitHub issues. When creating an enhancement suggestion, please include:

- **Use a clear and descriptive title**
- **Provide a detailed description of the proposed enhancement**
- **Explain why this enhancement would be useful**
- **List any alternative solutions you've considered**

Your First Code Contribution

Unsure where to begin? You can start by looking through these issues:

- `good first issue` - issues which should only require a few lines of code
- `help wanted` - issues which need extra attention
- `documentation` - issues related to documentation

Pull Requests

1. **Create a feature branch**:
   ```bash
   git checkout -b feature/your-feature-name
   ```
2. **Make your changes** following our style guidelines
3. **Write/update tests** for your changes
4. **Run tests and linters**:
   ```bash
   ```

```
make test
make lint
```

5. **Commit your changes** using our commit message conventions
6. **Push to your fork**:

   ```bash
   git push origin feature/your-feature-name
   ```

7. **Open a Pull Request** with a clear title and description

Development Process

Branching Strategy

We use Git Flow:

- `main` - stable release branch
- `develop` - integration branch for features
- `feature/*` - feature branches
- `bugfix/*` - bugfix branches
- `hotfix/*` - urgent fixes for production

Testing

- Write tests for all new functionality
- Maintain or increase code coverage
- All tests must pass before merging

```bash
# Run all tests
pytest

# Run with coverage
pytest --cov=src --cov-report=html

# Run specific test file
pytest tests/test_models.py

# Run tests in parallel
pytest -n auto
```

Code Quality

We use several tools to maintain code quality:

```bash
# Format code
black src/ tests/

# Sort imports
isort src/ tests/

# Lint code
flake8 src/ tests/

# Type checking
mypy src/

# Security checks
bandit -r src/

# All checks
make lint
```

Style Guidelines

Python Style Guide

We follow PEP 8 with some modifications:

- Line length: 100 characters
- Use Black for formatting
- Use isort for import sorting

Code Style

```python
```

```
# Good
def calculate_over25_probability(
    home_goals_avg: float,
    away_goals_avg: float,
    league_avg: float
) -> float:
    """
    Calculate probability of Over 2.5 goals.

    Args:
        home_goals_avg: Home team average goals
        away_goals_avg: Away team average goals
        league_avg: League average Over 2.5 rate

    Returns:
        Probability between 0 and 1
    """
    total_expected = home_goals_avg + away_goals_avg
    base_prob = total_expected / 5.0  # Normalize

    # Adjust for league tendency
    adjusted_prob = base_prob * 0.7 + league_avg * 0.3

    return min(max(adjusted_prob, 0.0), 1.0)
```

Documentation

- Use Google-style docstrings
- Document all public functions and classes
- Include type hints
- Add examples for complex functions

Testing Style

```
python
```

```python
# Good test example
import pytest
from src.models.predictor import Over25Predictor

class TestOver25Predictor:
    """Test cases for Over25Predictor."""

    @pytest.fixture
    def predictor(self):
        """Create predictor instance."""
        return Over25Predictor(model_type="random_forest")

    def test_predict_returns_valid_probability(self, predictor):
        """Test that predict returns probability between 0 and 1."""
        # Arrange
        features = {"home_goals_avg": 1.5, "away_goals_avg": 1.3}

        # Act
        result = predictor.predict(features)

        # Assert
        assert 0 <= result.probability <= 1
        assert result.confidence in ["High", "Medium", "Low"]
```

Commit Messages

We follow the Conventional Commits specification:

Format

```
<type>(<scope>): <subject>

<body>

<footer>
```

Types

- `feat`: New feature
- `fix`: Bug fix
- `docs`: Documentation changes
- `style`: Code style changes (formatting, etc.)
- `refactor`: Code refactoring

- `perf`: Performance improvements
- `test`: Adding or updating tests
- `build`: Build system changes
- `ci`: CI configuration changes
- `chore`: Other changes

Examples

```bash
# Good
git commit -m "feat(models): add XGBoost to ensemble model"
git commit -m "fix(api): handle rate limit errors gracefully"
git commit -m "docs: update installation instructions"

# Bad
git commit -m "fixed stuff"
git commit -m "WIP"
```

Pull Request Process

1. **Ensure all tests pass** and coverage hasn't decreased
2. **Update documentation** for any API changes
3. **Add yourself to CONTRIBUTORS.md** if this is your first contribution
4. **Fill out the PR template** completely
5. **Request review** from maintainers
6. **Address feedback** promptly
7. **Squash commits** before merging if requested

PR Template

```markdown
```

```
## Description
Brief description of changes

## Type of Change
- [ ] Bug fix
- [ ] New feature
- [ ] Breaking change
- [ ] Documentation update

## Testing
- [ ] All tests pass
- [ ] Added new tests
- [ ] Manual testing completed

## Checklist
- [ ] Code follows style guidelines
- [ ] Self-review completed
- [ ] Documentation updated
- [ ] No new warnings
```

Release Process

1. Update version in `setup.py` and `pyproject.toml`
2. Update CHANGELOG.md
3. Create release PR from `develop` to `main`
4. After merge, tag release: `git tag -a v2.1.0 -m "Release version 2.1.0"`
5. Push tag: `git push origin v2.1.0`
6. Create GitHub release
7. Deploy to production

Community

Getting Help

- 💬 Discord Server
- 📧 dev@profootball-ai.com
- 🐛 Issue Tracker

Weekly Meetings

We hold community calls every Thursday at 4 PM UTC. Join our Discord for details.

Recognition

Contributors are recognized in:

- CONTRIBUTORS.md file
- Release notes
- Annual contributor spotlight

License

By contributing, you agree that your contributions will be licensed under the MIT License.

Thank you for contributing to ProFootballAI! 🚀⚽