VNIT Housing Price Prediction Documentation

Release 1.0.0

Shivam Awasthi, Abhishiek Bhadauria

Dec 07, 2025

Contents

Welcome to the VNIT Housing Price Prediction project documentation. This comprehensive guide covers the complete machine learning pipeline including data preprocessing, exploratory data analysis, model optimization, and deployment.

1 Project Overview

The VNIT Housing Price Prediction project is an end-to-end machine learning solution that predicts house prices using advanced regression techniques. The project implements best practices in data science including:

- **Data Preprocessing & EDA** - Comprehensive data cleaning and exploration
- **Hyperparameter Optimization** - Optuna-based parameter tuning
- **Model Training** - XGBoost with cross-validation
- **Experiment Tracking** - MLflow for reproducibility
- **Performance Monitoring** - Evidently for model health checks
- **Workflow Orchestration** - Prefect for pipeline management

2 Key Metrics

- **Model R^2 Score**: 0.8990
- **RMSE**: $27,836.04
- **MAE**: $17,279.64
- **Optimization Trials**: 50 (Optuna)
- **Cross-Validation Folds**: 5

3 Quick Start

1. **Install Dependencies**:

```
pip install -r requirements.txt
```

2. **Run Preprocessing Pipeline**:

```
===============================================
VNIT Housing Price Prediction
===============================================


.. image:: https://img.shields.io/badge/Python-3.8+-blue.svg
.. image:: https://img.shields.io/badge/License-MIT-green.svg

Welcome to the VNIT Housing Price Prediction project documentation.␣
↪This comprehensive guide covers the complete machine learning␣
↪pipeline including data preprocessing, exploratory data analysis,␣
↪model optimization, and deployment.

Project Overview
================


The VNIT Housing Price Prediction project is an end-to-end machine␣
↪learning solution that predicts house prices using advanced␣
↪regression techniques. The project implements best practices in␣
↪data science including:

- **Data Preprocessing & EDA** - Comprehensive data cleaning and␣
↪exploration
- **Hyperparameter Optimization** - Optuna-based parameter tuning
- **Model Training** - XGBoost with cross-validation
```

(continues on next page)

(continued from previous page)

```
- **Experiment Tracking** - MLflow for reproducibility
- **Performance Monitoring** - Evidently for model health checks
- **Workflow Orchestration** - Prefect for pipeline management

Key Metrics
===========

- **Model R² Score**: 0.8990
- **RMSE**: $27,836.04
- **MAE**: $17,279.64
- **Optimization Trials**: 50 (Optuna)
- **Cross-Validation Folds**: 5

Quick Start
===========

1. **Install Dependencies**::

    pip install -r requirements.txt

2. **Run Preprocessing Pipeline**::

    cd src/preprocessing
    python preprocessing.py ../../data/train.csv

3. **Train Model**::

    cd src/modeling
    python modeling.py

4. **Launch Streamlit App**::

    streamlit run src/app/streamlit_app.py

Project Structure
=================

::

    VNIT_project/
    ├── data/                        # Dataset files
    │   ├── train.csv                # Training data
    │   ├── test.csv                 # Test data
```

(continues on next page)

(continued from previous page)

```
    │   ├── X_train.csv              # Preprocessed features
    │   └── data_description.txt     # Feature descriptions
    ├── models/                      # Trained models
    │   └── xgboost_model.joblib     # Serialized XGBoost model
    ├── src/
    │   ├── preprocessing/           # Data preprocessing modules
    │   │   ├── preprocessing.py     # Preprocessing pipeline
    │   │   ├── monitoring.py        # Data quality monitoring
    │   │   └── EDADataPreProcessing.ipynb
    │   ├── modeling/                # Model training modules
    │   │   ├── modeling.py          # Model pipeline
    │   │   └── Modeling.ipynb
    │   ├── app/                     # Streamlit application
    │   │   └── streamlit_app.py
    │   └── mlruns/                  # MLflow experiment tracking
    ├── docs/                        # Documentation
    ├── Dockerfile                   # Docker configuration
    ├── docker-compose.yml           # Docker Compose setup
    └── requirements.txt             # Python dependencies

Table of Contents
=================

.. toctree::
   :maxdepth: 2
   :caption: Getting Started

   getting_started
   installation
   quick_start

.. toctree::
   :maxdepth: 2
   :caption: Modules & APIs

   modules/preprocessing
   modules/monitoring
   modules/modeling

.. toctree::
   :maxdepth: 2
   :caption: Guides

```

(continues on next page)

(continued from previous page)

```
   guides/data_preprocessing
   guides/model_training
   guides/deployment

.. toctree::
   :maxdepth: 2
   :caption: Reference

   api_reference
   troubleshooting
   faq

Features
========

✓ **Automated Data Pipeline** - End-to-end preprocessing with Prefect
✓ **Hyperparameter Optimization** - Optuna integration for parameter␣
↪tuning
✓ **Experiment Tracking** - MLflow for model versioning and␣
↪comparison
✓ **Data Quality Monitoring** - Evidently for drift detection
✓ **Model Evaluation** - Comprehensive metrics and validation
✓ **Production Ready** - Docker support and API deployment
✓ **Documentation** - Complete Sphinx documentation with examples

Technologies Used
=================

- **ML Framework**: XGBoost
- **Hyperparameter Tuning**: Optuna
- **Experiment Tracking**: MLflow
- **Workflow Orchestration**: Prefect
- **Data Quality**: Evidently
- **Data Processing**: Pandas, NumPy, Scikit-learn
- **Visualization**: Matplotlib, Seaborn
- **Deployment**: Streamlit, Docker

Performance
===========

The final XGBoost model achieves excellent performance on the test␣
↪set:
```

(continues on next page)

(continued from previous page)

```
- **R² Score**: 0.8990 (explains 89.90% of variance)
- **Root Mean Squared Error**: $27,836
- **Mean Absolute Error**: $17,280
- **Training Samples**: 1,168
- **Test Samples**: 292

Contributing
============

Contributions are welcome! Please feel free to submit pull requests␣
↪or open issues for bugs and feature requests.

Authors
=======

- **Shivam Awasthi** (Data Science Engineer)
- **Abhishiek Bhadauria** (Data Science Engineer)

License
=======

This project is licensed under the MIT License - see the LICENSE␣
↪file for details.

Acknowledgments
===============

- Kaggle Housing Dataset
- Scikit-learn and XGBoost communities
- Optuna optimization framework
- MLflow experiment tracking
- Evidently ML model monitoring

Contact
=======

For questions or support, please open an issue on GitHub or contact␣
↪the maintainers.

---

.. note::
```

(continues on next page)

(continued from previous page)

```
   This documentation is comprehensive and up-to-date as of December␣
↪2025. For the latest updates, please check the GitHub repository.

Last Updated: December 6, 2025
```