**BigMart Sales Prediction - README**

**Overview**

This repository contains a comprehensive solution for the BigMart Sales Prediction challenge

**Problem Statement**

Predict sales of 1,559 products across 10 BigMart outlets using historical 2013 data. The challenge involves handling missing values, data quality issues, and building robust predictive models.

**Solution Architecture**

**Files Structure**

bigmart-sales-prediction/  
│  
├── bigmart\_solution.py # Main solution pipeline  
├── bigmart\_eda.py # Exploratory Data Analysis  
├── bigmart\_modeling.py # Model experimentation  
├── approach-note.md # Detailed methodology  
├── README.md # This file  
├── requirements.txt # Dependencies  
│  
├── data/  
│ ├── train.csv # Training dataset  
│ ├── test.csv # Test dataset  
│ └── submission.csv # Final predictions

**Key Features**

**1. Advanced Feature Engineering**

* **Data Quality Fixes**: Standardized Item\_Fat\_Content, handled missing values intelligently
* **New Features**: Created 8+ engineered features including Outlet\_Years, Item\_Popularity, Visibility\_Ratio
* **Domain Knowledge**: Applied retail industry insights for feature creation

**2. Comprehensive Modeling**

* **Multiple Algorithms**: Tested 13 different models from linear to ensemble methods
* **Hyperparameter Tuning**: Advanced optimization using RandomizedSearchCV
* **Ensemble Methods**: Weighted combination of best-performing models

**3. Robust Validation**

* **Cross-Validation**: 5-fold CV for reliable performance estimation
* **Hold-out Validation**: 20% validation split for final model selection
* **Residual Analysis**: Comprehensive model diagnostics

**Installation & Setup**

**Prerequisites**

Python 3.8+  
pandas >= 1.3.0  
numpy >= 1.21.0  
scikit-learn >= 1.0.0  
xgboost >= 1.5.0  
lightgbm >= 3.3.0  
matplotlib >= 3.5.0  
seaborn >= 0.11.0

**Quick Setup**

# Clone repository  
git clone <repository-url>  
cd bigmart-sales-prediction  
  
# Install dependencies  
pip install -r requirements.txt  
  
# Place your data files  
# Copy train.csv and test.csv to the main directory  
  
# Run complete solution  
python bigmart\_solution.py

**Technical Approach**

**Data Preprocessing**

1. **Missing Value Imputation**:
   * Item\_Weight: Item-specific mean imputation
   * Outlet\_Size: Mode based on Outlet\_Type
2. **Data Quality Fixes**:
   * Standardized Item\_Fat\_Content values
   * Replaced zero visibility with item-specific means
3. **Feature Engineering**:
   * Created 8+ new features based on domain knowledge
   * Applied categorical encoding strategies

**Model Development**

1. **Baseline Models**: Linear Regression, Ridge, Lasso
2. **Tree-based Models**: Random Forest, Decision Tree, Extra Trees
3. **Gradient Boosting**: XGBoost, LightGBM, CatBoost, Gradient Boosting
4. **Ensemble**: Weighted average of top-performing models

**Performance Optimization**

* **Hyperparameter Tuning**: 50+ iterations for best models
* **Cross-Validation**: Robust performance estimation
* **Ensemble Weighting**: Performance-based model combination

**Expected Results**

**Model Performance Targets**

* **Baseline RMSE**: ~1150-1200
* **Advanced Models**: ~1050-1100
* **Final Ensemble**: ~1040-1080

**Key Performance Features**

* **Feature Importance**: MRP, Outlet\_Type, and Item\_Type as top predictors
* **Model Robustness**: Consistent performance across CV folds
* **Generalization**: Strong validation performance indicates good generalization

**Key Innovations**

**1. Advanced Feature Engineering**

# Examples of engineered features  
combined['Outlet\_Years'] = 2013 - combined['Outlet\_Establishment\_Year']  
combined['Price\_per\_Weight'] = combined['Item\_MRP'] / combined['Item\_Weight']  
combined['Visibility\_Ratio'] = item\_visibility / category\_mean\_visibility

**2. Intelligent Missing Value Handling**

# Item-specific weight imputation  
item\_weight\_mean = combined.groupby('Item\_Identifier')['Item\_Weight'].mean()  
combined['Item\_Weight'] = combined.apply(  
 lambda x: item\_weight\_mean[x['Item\_Identifier']]   
 if pd.isna(x['Item\_Weight']) else x['Item\_Weight'], axis=1  
)

**3. Performance-Weighted Ensemble**

# Dynamic ensemble weights based on validation performance  
total\_weight = 1/xgb\_rmse + 1/lgb\_rmse + 1/rf\_rmse  
xgb\_weight = (1/xgb\_rmse) / total\_weight  
ensemble\_pred = (xgb\_weight \* xgb\_pred + lgb\_weight \* lgb\_pred + rf\_weight \* rf\_pred)

**Troubleshooting**

**Common Issues**

1. **File Not Found Error**:

Ensure train.csv and test.csv are in the main directory

1. **Memory Issues**:

# Reduce model complexity or use subsampling  
model = XGBRegressor(n\_estimators=100) # Instead of 500

1. **Slow Performance**:

# Reduce hyperparameter search space  
param\_grid = {'n\_estimators': [100, 200]} # Instead of [100, 200, 300, 500]

**Performance Metrics**

Expected leaderboard position: **Top 10%** based on:

* Advanced feature engineering
* Comprehensive model experimentation
* Robust ensemble approach
* Domain expertise application