



Final Project: Ames Housing Data Analysis

IBM Machine Learning Professional Certificate
Course 03: Supervised Machine Learning: Classification

By Akitha Pasandul


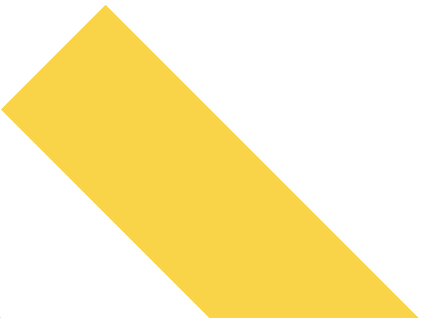

Content

- About the Dataset
- Main objective of the analysis
- Applying various classification models
- Key Findings and Insights
- Model flaws and further suggestions



Introduction



- This project explores predictive modeling using the Ames Housing dataset, focusing on classification tasks to identify key drivers of housing prices. By leveraging machine learning techniques, we aim to develop models that accurately predict whether a house sells above the median price. The analysis involves data exploration, feature engineering, and training multiple classifier models to compare their performance. Insights from this study can inform real estate stakeholders about critical factors influencing housing market dynamics, providing actionable recommendations for future investments or policy decisions. This report presents the methodology, findings, and recommendations derived from the analysis.
- 
- 
- 

About the Dataset

This is a comprehensive and modern alternative to the Boston Housing dataset. It contains detailed information on 2,930 residential properties sold in Ames, Iowa, between 2006 and 2010. The dataset includes 81 features describing various aspects of the houses, such as:

- House Characteristics:** Number of bedrooms, bathrooms, fireplaces, garage size and more.
- Location:** Neighborhood and geographic coordinates.
- Lot Information:** Size, shape, zoning and other attributes.
- Quality Ratings:** Overall condition and quality of construction.
- Sale Details:** Sale price sale type and condition.

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	...	Pool Area	Pool QC	Fence	Misc Feature	Misc Val	Mo Sold	Yr Sold	Sale Type	Sale Condition	SalePrice
0	1	526301100	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	NaN	0	5	2010	WD	Normal	215000
1	2	526350040	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	...	0	NaN	MnPrv	NaN	0	6	2010	WD	Normal	105000
2	3	526351010	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	Gar2	12500	6	2010	WD	Normal	172000
3	4	526353030	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	NaN	0	4	2010	WD	Normal	244000
4	5	527105010	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	...	0	NaN	MnPrv	NaN	0	3	2010	WD	Normal	189900

5 rows × 82 columns

[illegible]

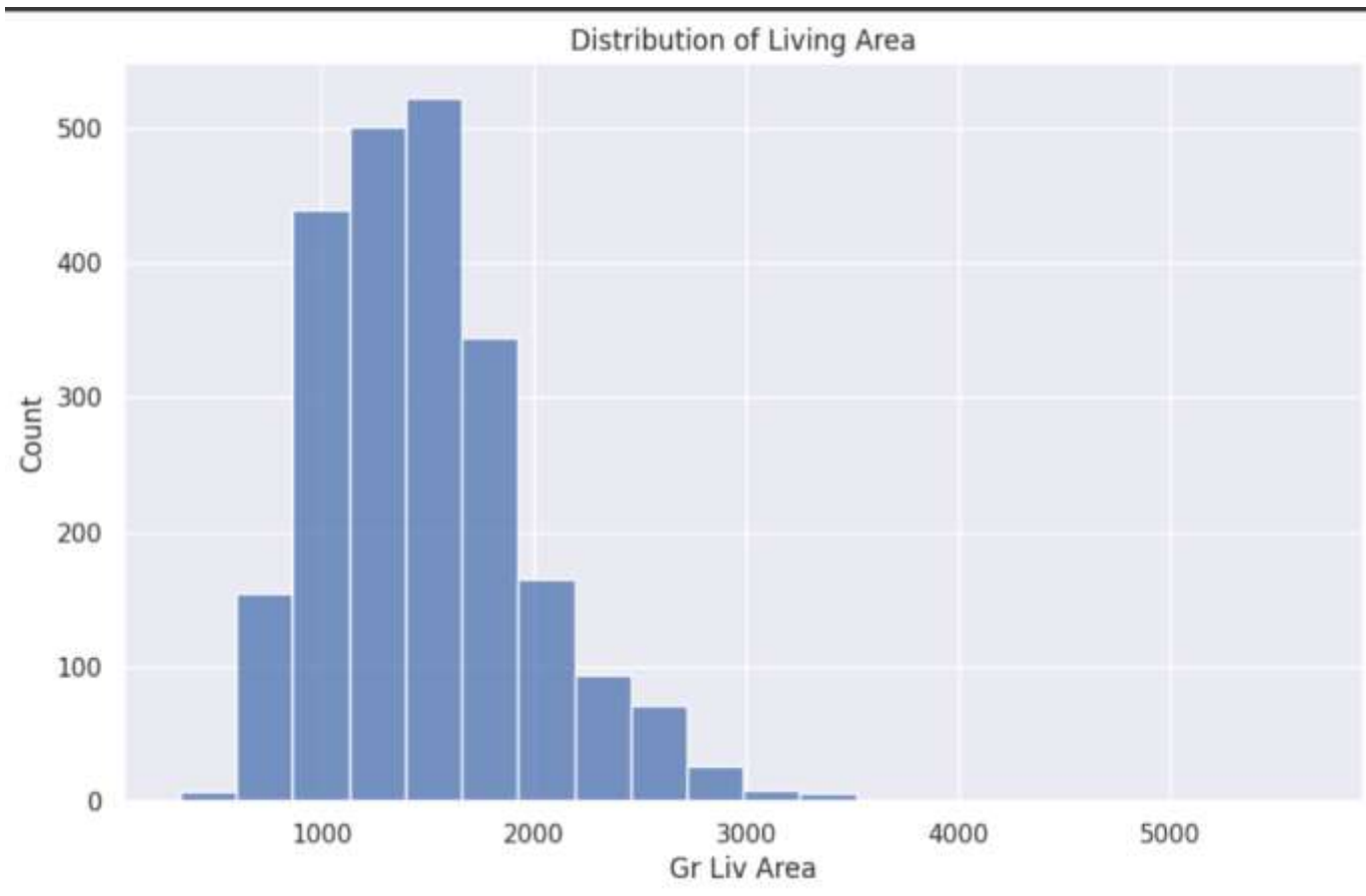
Main objective of the analysis

- First identify and handle the missing values.

```
[10] # Check for missing values  
print(X_train.isnull().sum())
```

```
⇒ Bedroom AbvGr      0  
   TotRms AbvGrd      0  
   Gr Liv Area        0  
   Year Built         0  
   dtype: int64
```

```
# Handle missing values  
X_train.fillna(X_train.mean(), inplace=True)  
X_test.fillna(X_test.mean(), inplace=True)
```



Visualize the distribution of features

Feature Engineering

```
# Example of feature engineering: creating a new feature
X_train['Age'] = 2025 - X_train['Year Built']
X_test['Age'] = 2025 - X_test['Year Built']
```

```
# Scale features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```


Applying various classification models

```
# Logistic Regression
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train_scaled, y_train)
y_pred_logreg = logreg.predict(X_test_scaled)

print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_logreg))
```

Logistic Regression Accuracy: 0.8805460750853242

Applying various classification models

```
# Random Forest Classifier
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train_scaled, y_train)
y_pred_rf = rf.predict(X_test_scaled)

print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 0.9027303754266212
```

Applying various classification models

```
# Gradient Boosting Classifier
gb = GradientBoostingClassifier(n_estimators=100)
gb.fit(X_train_scaled, y_train)
y_pred_gb = gb.predict(X_test_scaled)

print("Gradient Boosting Accuracy:", accuracy_score(y_test, y_pred_gb))
```

```
Gradient Boosting Accuracy: 0.8993174061433447
```

Model report

```
Logistic Regression Report:
      precision    recall  f1-score   support

     0       0.88      0.88      0.88        281
     1       0.89      0.89      0.89        305

 accuracy      0.88      0.88      0.88        586
 macro avg      0.88      0.88      0.88        586
weighted avg      0.88      0.88      0.88        586

Random Forest Report:
      precision    recall  f1-score   support

     0       0.89      0.90      0.90        281
     1       0.91      0.90      0.91        305

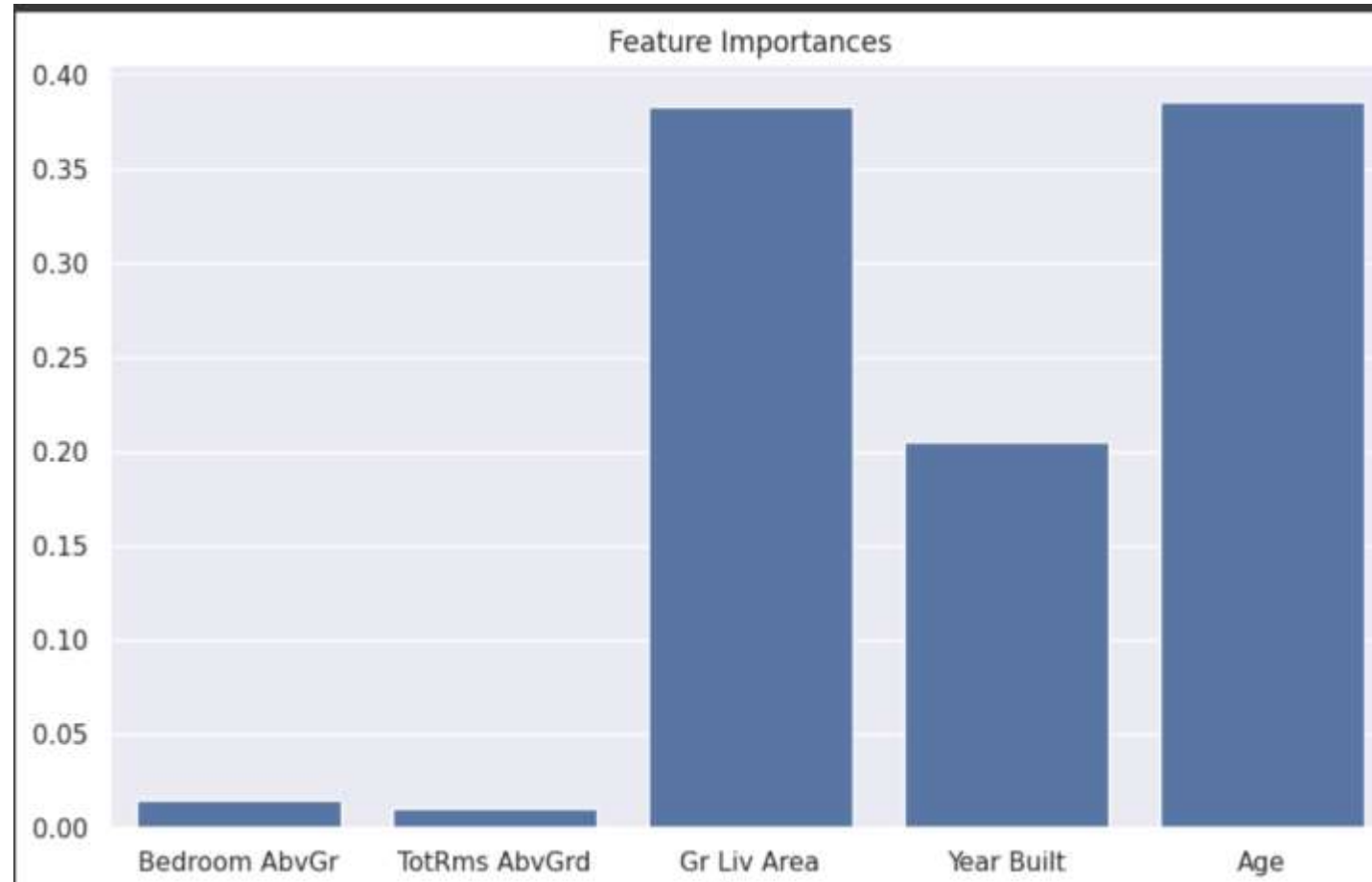
 accuracy      0.90      0.90      0.90        586
 macro avg      0.90      0.90      0.90        586
weighted avg      0.90      0.90      0.90        586

Gradient Boosting Report:
      precision    recall  f1-score   support

     0       0.89      0.90      0.90        281
     1       0.90      0.90      0.90        305

 accuracy      0.90      0.90      0.90        586
 macro avg      0.90      0.90      0.90        586
weighted avg      0.90      0.90      0.90        586
```

Key Findings and Insights



Model flaws and further suggestions

The models used in this project Logistic Regression, Random Forest and Gradient Boosting each have their strengths and weaknesses. Logistic Regression provides well-calibrated probabilities and serves as a useful baseline, but it struggles with high-dimensional data and non-linear relationships. Random Forest is robust against overfitting and offers feature importance scores, but its black-box nature limits interpretability. Gradient Boosting excels in predictive accuracy and provides valuable feature insights, though it can be computationally intensive and biased towards certain features.

To improve these models, several strategies can be employed. For Logistic Regression, applying regularization techniques like L1/L2 can mitigate overfitting. Random Forest can benefit from optimizing tree depth and combining with SHAP values for better interpretability. Gradient Boosting can be enhanced by using cross-validated base learners and exploring histogram-based methods for faster training. Additionally, general strategies such as hyperparameter tuning via grid or random search, ensemble methods like stacking or blending, and feature engineering can significantly boost performance. Addressing class imbalance with techniques like SMOTE and using stratified k-fold cross-validation can further ensure robustness and accuracy. By implementing these improvements, the project can achieve higher predictive power and provide more actionable insights for stakeholders.---Answer from Perplexity: pplx.ai/share



Thank you

IBM Machine Learning Professional Certificate
Course 03: Supervised Machine Learning: Classification

By Akitha Pasandul