# Datasheet for dataset “The Ames Housing”

Questions from the [Datasheets for Datasets](https://arxiv.org/abs/1803.09010) paper, v7.

Sections:

* [Motivation](#motivation)
* [Composition](#composition)
* [Collection process](#collection-process)
* [Preprocessing/cleaning/labeling](#preprocessingcleaninglabeling)
* [Uses](#uses)
* [Distribution](#distribution)
* [Maintenance](#maintenance)

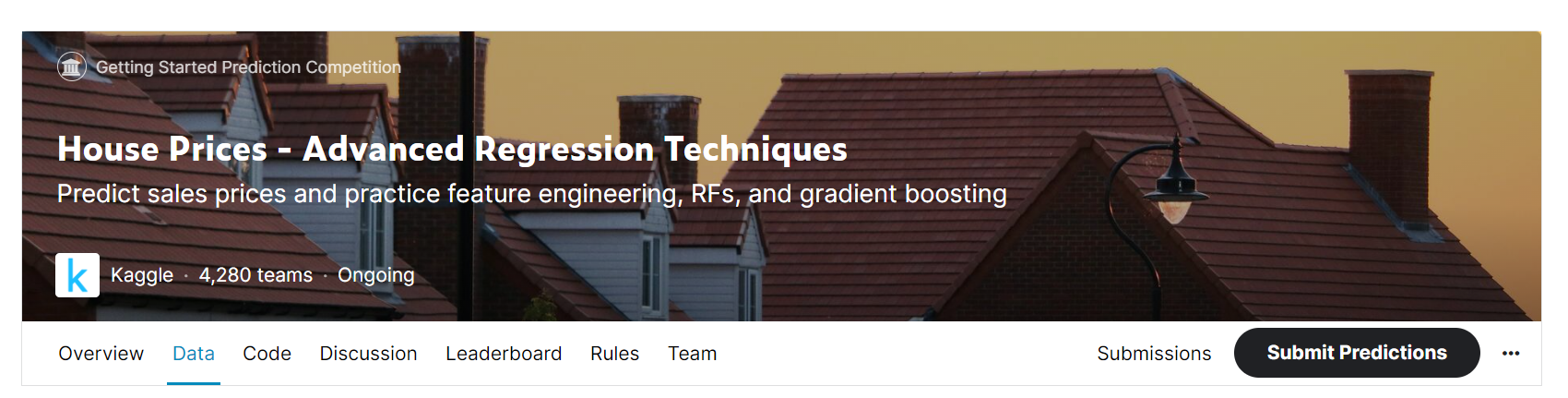
## Motivation

### 1- For what purpose was the dataset created?

### 2 - Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?

The Ames Housing dataset was compiled by Dean De Cock for use in data science education.

Link to the dataset: <https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview>



## Composition

### 4 - What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)?

### There are 79 explanatory variables describing aspects of residential homes in Ames, Iowa.

### Data fields:

### SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.

### MSSubClass: The building class

### MSZoning: The general zoning classification

### LotFrontage: Linear feet of street connected to property

### LotArea: Lot size in square feet

### Street: Type of road access

### Alley: Type of alley access

### LotShape: General shape of property

### LandContour: Flatness of the property

### Utilities: Type of utilities available

### LotConfig: Lot configuration

### LandSlope: Slope of property

### Neighborhood: Physical locations within Ames city limits

### Condition1: Proximity to main road or railroad

### Condition2: Proximity to main road or railroad (if a second is present)

### BldgType: Type of dwelling

### HouseStyle: Style of dwelling

### OverallQual: Overall material and finish quality

### OverallCond: Overall condition rating

### YearBuilt: Original construction date

### YearRemodAdd: Remodel date

### RoofStyle: Type of roof

### RoofMatl: Roof material

### Exterior1st: Exterior covering on house

### Exterior2nd: Exterior covering on house (if more than one material)

### MasVnrType: Masonry veneer type

### MasVnrArea: Masonry veneer area in square feet

### ExterQual: Exterior material quality

### ExterCond: Present condition of the material on the exterior

### Foundation: Type of foundation

### BsmtQual: Height of the basement

### BsmtCond: General condition of the basement

### BsmtExposure: Walkout or garden level basement walls

### BsmtFinType1: Quality of basement finished area

### BsmtFinSF1: Type 1 finished square feet

### BsmtFinType2: Quality of second finished area (if present)

### BsmtFinSF2: Type 2 finished square feet

### BsmtUnfSF: Unfinished square feet of basement area

### TotalBsmtSF: Total square feet of basement area

### Heating: Type of heating

### HeatingQC: Heating quality and condition

### CentralAir: Central air conditioning

### Electrical: Electrical system

### 1stFlrSF: First Floor square feet

### 2ndFlrSF: Second floor square feet

### LowQualFinSF: Low quality finished square feet (all floors)

### GrLivArea: Above grade (ground) living area square feet

### BsmtFullBath: Basement full bathrooms

### BsmtHalfBath: Basement half bathrooms

### FullBath: Full bathrooms above grade

### HalfBath: Half baths above grade

### Bedroom: Number of bedrooms above basement level

### Kitchen: Number of kitchens

### KitchenQual: Kitchen quality

### TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

### Functional: Home functionality rating

### Fireplaces: Number of fireplaces

### FireplaceQu: Fireplace quality

### GarageType: Garage location

### GarageYrBlt: Year garage was built

### GarageFinish: Interior finish of the garage

### GarageCars: Size of garage in car capacity

### GarageArea: Size of garage in square feet

### GarageQual: Garage quality

### GarageCond: Garage condition

### PavedDrive: Paved driveway

### WoodDeckSF: Wood deck area in square feet

### OpenPorchSF: Open porch area in square feet

### EnclosedPorch: Enclosed porch area in square feet

### 3SsnPorch: Three season porch area in square feet

### ScreenPorch: Screen porch area in square feet

### PoolArea: Pool area in square feet

### PoolQC: Pool quality

### Fence: Fence quality

### MiscFeature: Miscellaneous feature not covered in other categories

### MiscVal: $Value of miscellaneous feature

### MoSold: Month Sold

### YrSold: Year Sold

### SaleType: Type of sale

### SaleCondition: Condition of sale

### 5 - How many instances are there in total (of each type, if appropriate)?

RangeIndex: 1460 entries, 0 to 1459

Data columns (total 81 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Id 1460 non-null int64

1 MSSubClass 1460 non-null int64

2 MSZoning 1460 non-null object

3 LotFrontage 1201 non-null float64

4 LotArea 1460 non-null int64

5 Street 1460 non-null object

6 Alley 91 non-null object

7 LotShape 1460 non-null object

8 LandContour 1460 non-null object

9 Utilities 1460 non-null object

10 LotConfig 1460 non-null object

11 LandSlope 1460 non-null object

12 Neighborhood 1460 non-null object

13 Condition1 1460 non-null object

14 Condition2 1460 non-null object

15 BldgType 1460 non-null object

16 HouseStyle 1460 non-null object

17 OverallQual 1460 non-null int64

18 OverallCond 1460 non-null int64

19 YearBuilt 1460 non-null int64

20 YearRemodAdd 1460 non-null int64

21 RoofStyle 1460 non-null object

22 RoofMatl 1460 non-null object

23 Exterior1st 1460 non-null object

24 Exterior2nd 1460 non-null object

25 MasVnrType 1452 non-null object

26 MasVnrArea 1452 non-null float64

27 ExterQual 1460 non-null object

28 ExterCond 1460 non-null object

29 Foundation 1460 non-null object

30 BsmtQual 1423 non-null object

31 BsmtCond 1423 non-null object

32 BsmtExposure 1422 non-null object

33 BsmtFinType1 1423 non-null object

34 BsmtFinSF1 1460 non-null int64

35 BsmtFinType2 1422 non-null object

36 BsmtFinSF2 1460 non-null int64

37 BsmtUnfSF 1460 non-null int64

38 TotalBsmtSF 1460 non-null int64

39 Heating 1460 non-null object

40 HeatingQC 1460 non-null object

41 CentralAir 1460 non-null object

42 Electrical 1459 non-null object

43 1stFlrSF 1460 non-null int64

44 2ndFlrSF 1460 non-null int64

45 LowQualFinSF 1460 non-null int64

46 GrLivArea 1460 non-null int64

47 BsmtFullBath 1460 non-null int64

48 BsmtHalfBath 1460 non-null int64

49 FullBath 1460 non-null int64

50 HalfBath 1460 non-null int64

51 BedroomAbvGr 1460 non-null int64

52 KitchenAbvGr 1460 non-null int64

53 KitchenQual 1460 non-null object

54 TotRmsAbvGrd 1460 non-null int64

55 Functional 1460 non-null object

56 Fireplaces 1460 non-null int64

57 FireplaceQu 770 non-null object

58 GarageType 1379 non-null object

59 GarageYrBlt 1379 non-null float64

60 GarageFinish 1379 non-null object

61 GarageCars 1460 non-null int64

62 GarageArea 1460 non-null int64

63 GarageQual 1379 non-null object

64 GarageCond 1379 non-null object

65 PavedDrive 1460 non-null object

66 WoodDeckSF 1460 non-null int64

67 OpenPorchSF 1460 non-null int64

68 EnclosedPorch 1460 non-null int64

69 3SsnPorch 1460 non-null int64

70 ScreenPorch 1460 non-null int64

71 PoolArea 1460 non-null int64

72 PoolQC 7 non-null object

73 Fence 281 non-null object

74 MiscFeature 54 non-null object

75 MiscVal 1460 non-null int64

76 MoSold 1460 non-null int64

77 YrSold 1460 non-null int64

78 SaleType 1460 non-null object

79 SaleCondition 1460 non-null object

80 SalePrice 1460 non-null int64

dtypes: float64(3), int64(35), object(43)

### 6 - Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?

### The dataset contain all possible instances.

### 7 - What data does each instance consist of?

Please refer to question number 4.

### 8 - Is there a label or target associated with each instance?

SalePrice - the property's sale price in dollars. This is the target.

### 9 - Is any information missing from individual instances?

Features with missing values:

Features Missing values

PoolQC 1453

MiscFeature 1406

Alley 1369

Fence 1179

FireplaceQu 690

LotFrontage 259

GarageYrBlt 81

GarageCond 81

GarageType 81

GarageFinish 81

GarageQual 81

BsmtFinType2 38

BsmtExposure 38

BsmtQual 37

BsmtCond 37

BsmtFinType1 37

MasVnrArea 8

MasVnrType 8

Electrical 1

Id 0

Functional 0

Fireplaces 0

For more details, please refer to question number 5.

### 10 - Are relationships between individual instances made explicit (e.g., users’ movie ratings, social network links)?

Yes, Street Name and Neighborhood are available.

### 11 - Are there recommended data splits (e.g., training, development/validation, testing)?

The data set split will be 70% for training and 30% for testing in order to evaluate our models.

### 12 - Are there any errors, sources of noise, or redundancies in the dataset?

* At first glance out of 1460 entries there are no identical rows.
* Some features have a high % of missing values.
* Some features are highly correlated each other.
* Some features are low correlated with the label.

### 13 - Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals’ non-public communications)?

Sensitive features related with Street Name and Neighborhood could be considered confidential but they are necessary for the model to predict the sale price of the house.

### 14 - Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?

No, it doesn’t contain offensive data.

### 15 - Does the dataset relate to people?

No, it’s related with houses, its features and sale price.

## Collection process

### 

### 16 - How was the data associated with each instance acquired?

### Information not available.

### 17 - What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)?

### Information not available.

### 18 - Over what timeframe was the data collected?

### Information not available.

## Cleaning

### 20 - Was any cleaning of the data done?

The data cleaning/preprocessing process was separated in 4 version

On this data cleaning process there are 4 versions of cleaning/preprocessing tasks.

On **version 0**(zero) was performed the minimum manipulation necessary to run a basic regression model, which will be use as base line model to compare.

**Version 1**, add to version 0 the changes proposed on version 1.

**Version 2**, add to version 0 the changes proposed on version 2.

**Version 3**, add to version 0 a combination of version 1 and 2.

The following changes or imputations were done on the **train** dataset **version 0**:

* Check for duplicates values.
* Drop features with more than 40% of missing values. The features to drop were 'Alley','PoolQC','Fence','MiscFeature'. Code:

df.drop(['Alley','PoolQC','Fence','MiscFeature'],axis=1,inplace=True)

* According to the database description, the 'nan' values in some features are due to the non-existence of the feature on that particular instance. Those missing values were filled with 'None' on the following features: 'BsmtFinType1','BsmtFinType2','BsmtQual','BsmtCond','BsmtExposure','FireplaceQu', 'MasVnrType','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond'. Code:

df[['BsmtFinType1','BsmtFinType2','BsmtQual','BsmtCond','BsmtExposure','FireplaceQu','MasVnrType','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']] = df[['BsmtFinType1','BsmtFinType2','BsmtQual','BsmtCond','BsmtExposure','FireplaceQu','MasVnrType','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']].fillna('None')

* Imputing the mode on missing value for the feature 'Electrical'. Code: df['Electrical'] = df['Electrical'].fillna('SBrkr')
* Imputing the mode of 'GarageYrBlt' for the top 3 'Neighborhood' where ['GarageYrBlt'].isnull(). Code: df[(df['Neighborhood']== 'Edwards') | (df['Neighborhood']== 'OldTown') | (df['Neighborhood']=='BrkSide')]['GarageYrBlt'].value\_counts()

df['GarageYrBlt'] = df['GarageYrBlt'].fillna(1950.0)

* Impute 0.0 on 'MasVnrArea' for the missing values base on the feature 'MasVnrType'. Code: df['MasVnrArea'] = df['MasVnrArea'].fillna(0.0)
* Impute missing numerical values on the feature 'LotFrontage' using IterativeImputer. IterativeImputer is a multivariate imputer that estimates each feature from all the others. It is useful to predict missing values on numerical features.

Link: [sklearn.impute.IterativeImputer — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html)

* One hot encoding the categorical features using OneHotEncoder (OneHotEncoder is a sklearn transformer that encode categorical features as a one-hot numeric array).

Link: [sklearn.preprocessing.OneHotEncoder — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html#sklearn.preprocessing.OneHotEncoder)

The following changes or imputations were done on the **test** dataset on **version 0**:

* Check for duplicates values.
* Drop features with more than 40% of missing values. The features to drop were 'Alley','PoolQC','Fence','MiscFeature'. Code:

df.drop(['Alley','PoolQC','Fence','MiscFeature'],axis=1,inplace=True)

* According to the database description, the 'nan' values in some features are due to the non-existence of the feature on that particular instance. Those missing values were filled with 'None' on the following features: 'BsmtFinType1','BsmtFinType2','BsmtQual','BsmtCond','BsmtExposure','FireplaceQu', 'MasVnrType','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond'. Code:

df[['BsmtFinType1','BsmtFinType2','BsmtQual','BsmtCond','BsmtExposure','FireplaceQu','MasVnrType','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']] = df[['BsmtFinType1','BsmtFinType2','BsmtQual','BsmtCond','BsmtExposure','FireplaceQu','MasVnrType','GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']].fillna('None').

* Filling missing values on the feature 'MSZoning' based on the mode. Code: dft['MSZoning'] = dft['MSZoning'].fillna('RL')
* Filling missing values on the feature 'Utilities' based on the mode. Code: dft['Utilities'] = dft['Utilities'].fillna('AllPub')
* Filling missing values on the feature 'Functional' based on the mode. Code: dft['Functional'] = dft['Functional'].fillna('Typ')
* Filling missing values on the feature 'KitchenQual' based on the mode considering the feature 'Neighborhood'. Code: dft['KitchenQual'] = dft['KitchenQual'].fillna('TA')
* Filling missing values on the feature 'SaleType' based on the mode. Code: dft['SaleType'] = dft['SaleType'].fillna('WD')
* Filling missing values on the feature 'Exterior2nd' based on the mode considering the feature 'Neighborhood'. Code: dft['Exterior2nd'] = dft['Exterior2nd'].fillna('Wd Sdng')
* Filling missing values on the feature 'Exterior1st' based on the mode considering the feature 'Neighborhood'. Code: dft['Exterior1st'] = dft['Exterior1st'].fillna('Wd Sdng')
* According to the database description, the 'nan' values in some features are due to the non-existence of the feature on that particular instance. Those missing values were filled with 0.0 on the following features: 'MasVnrArea','BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','GarageCars','GarageArea','BsmtFullBath','BsmtHalfBath'.Code: dft[['MasVnrArea','BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','GarageCars','GarageArea','BsmtFullBath','BsmtHalfBath']] = dft[['MasVnrArea','BsmtFinSF1','BsmtFinSF2','BsmtUnfSF','TotalBsmtSF','GarageCars','GarageArea','BsmtFullBath','BsmtHalfBath']].fillna(0.0)
* Filling missing values on the feature 'GarageYrBlt' based on the feature 'GarageCond. Code: dft['GarageYrBlt'] = dft['GarageYrBlt'].fillna(0.0)
* Impute missing numerical values on the feature 'LotFrontage' using IterativeImputer. IterativeImputer is a multivariate imputer that estimates each feature from all the others. It is useful to predict missing values on numerical features.

Link: [sklearn.impute.IterativeImputer — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html)

* One hot encoding the categorical features using OneHotEncoder (OneHotEncoder is a sklearn transformer that encode categorical features as a one-hot numeric array).

Link: [sklearn.preprocessing.OneHotEncoder — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html#sklearn.preprocessing.OneHotEncoder)

* Standardization was applied on train and test set to fit some of the models created. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using transform (Sklearn documentation).

Link: [6.3. Preprocessing data — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/modules/preprocessing.html)

* Changes proposed in **version 1** for both, train and test set:

Drop features high correlated between each other.

Drop features low correlated with the label.

* Changes proposed in **version 2** for both, train and test set:

Replace outliers (on features highly correlated with the label correlation >=|0.60|) for the mean value of the feature

* Features distribution with heavy tail or skewed were transformed using np.sqrt() to make the distribution roughly symmetrical.
* Changes proposed in **version 3** are a combination of version 1 and 2 with the difference that outliers on train dataset were deleted meanwhile the outliers on the test set were replaced for the mean value of the feature (it is a Kaggle submission requirement to not alter the number of rows on the test set).

### Was the “raw” data saved in addition to the cleaned data (e.g., to support unanticipated future uses)?

Yes using the method .copy()

## Uses

The Ames Housing dataset was created for educational purposes.

### Has the dataset been used for any tasks already?

Yes, it is used in Kaggle competitions for educational purposes.

### Is there a repository that links to any or all papers or systems that use the dataset?

### Information about Ames Housing dataset can be found at the following link:

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

### Is there anything about the composition of the dataset or the way it was collected and cleaned that might impact future uses?

Information not available.

### Are there tasks for which the dataset should not be used?

Information not available.

## Distribution

### Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created?

The database is publicly accessible, for educational purposes.

### How will the dataset will be distributed (e.g., zip file, website, GitHub)?

Once the data cleaning process is done, the dataset will be uploaded to GitHub.

### Who is supporting/hosting/maintaining the dataset?

The dataset is hosting by Kaggle.com

### How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

The owner could be contacted through Kaggle.com

### Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)?

Information not available.

### If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?

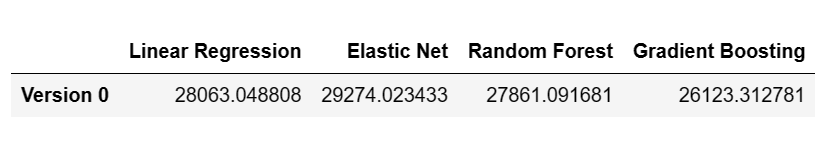
Information not available. Refer to Kaggle.com

## Models performance:

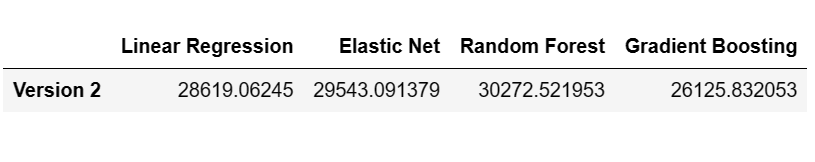
Since this is a regression problem where, given a set of features, the goal is to predict the sale value of a house. To achieve that goal 4 estimators on each version will be tested, these are: LinearRegression, ElasticNet, RandomForestRegressor, GradientBoostingRegressor.

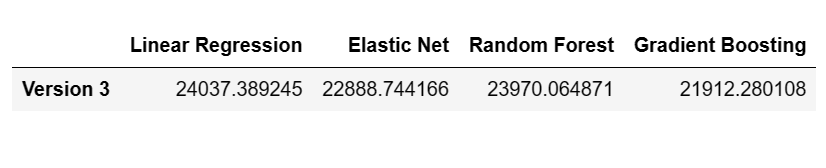
The metric used to evaluate the performance was **Root** **Mean Squared Error** **(RMSE).**

## Models performance on version 0 (Base line model):



**Models performance on version 1:**

**Models performance on version 2:**

**Models performance on version 3:**

**Conclusion:**

The best performance on the training set was achieved in version 3 by the gradient boosting regressor estimator.

## Kaggle performance:

Testing the best performance of each version on Kaggle, the best result was again obtained by the gradient boosting regressor estimator on version 3.

