Ames Housing Dataset

By Random Forest Rangers:
Dmitry Gufranov, Evgenia Amineva, Valeriya Vazhnova

Objectives of the Research

- Study Ames housing dataset
- Predict sale price of the property given 79 explanatory variables
- Apply theoretical knowledge from YData on a real-world problem
- Try fancy regression models, i.e. CatBoost or XGBoost
- Practice teamwork

About the City

Ames is a city in Iowa, best known as the home for Iowa State University (ISU). Ranks as ninth most populous city in Iowa. Students make up almost half of its population.

- Founded: 1864
- Population: 66,427 (according to 2020 census)
- Area: 27.92 sq miles (72.32 sq. km)
- Climate: humid continental

Work Process

Our work on the task was held in several stages:

1. Initial preprocessing
2. Training baseline model
3. Deep preprocessing and feature engineering
4. Applying nonlinear algorithms
5. Building stacked model
6. Hyperparameter optimization (GridSearchCV, Optuna)

Preprocessing

Structure of the dataset

Train	Test
1460 observations with SalesPrice (IDs 1-1460)	1459 observations (IDs 1461-2919)
79 features	79 features
No duplicates	No duplicates
	Target: SalesPrice

Missing Values

It is established that our dataset has missing values. In some cases that genuinely means that the data is not provided, while in some cases it's an indicator of a certain condition (i.e. missing values in **PoolQC** mean that the property has no pool, missing values in **Alley** mean no alley etc.).

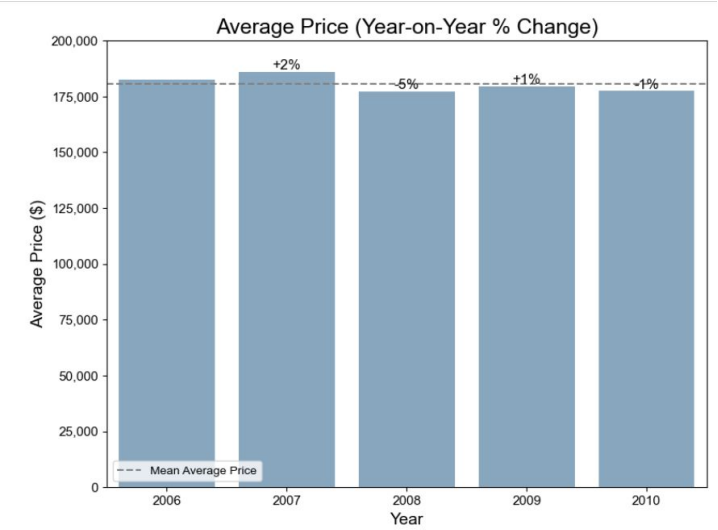
In dealing with missing values we've adopted a following strategy, dividing columns with missing values into several categories:

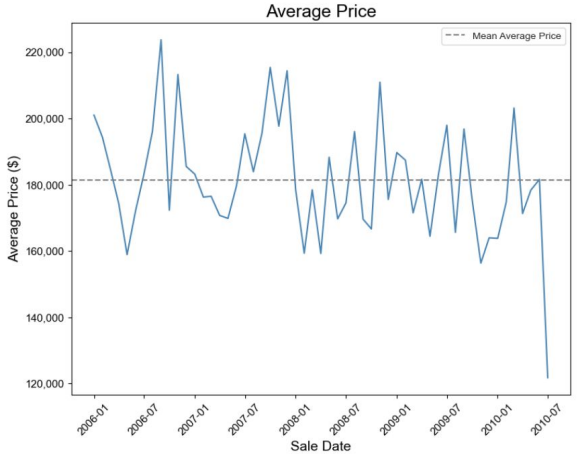
- **Categorical features.** Creating a new category: NA (e.g. in PoolQC or GarageType)
- **Categorical features.** Filling with most frequent values
- **Numerical features.** Filling with 0 (e.g. in GarageArea for properties with no garage)
- **Numerical features.** Filling with average values / average per category (i.e. average LotFrontage per neighborhood).

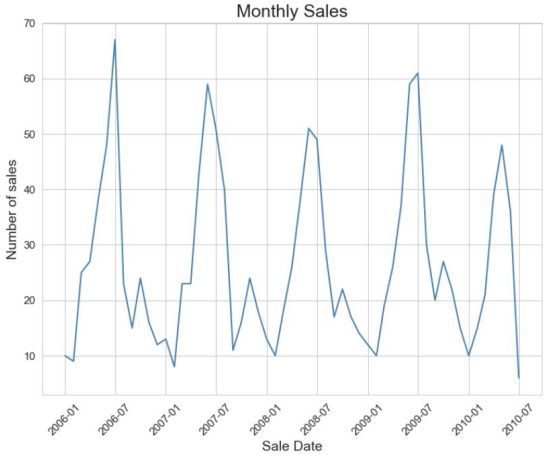
```
Percantage of missing values by columns:

PoolQC          99.520548
MiscFeature     96.301370
Alley           93.767123
Fence           80.753425
MasVnrType      59.726027
FireplaceQu     47.260274
LotFrontage     17.739726
GarageType       5.547945
GarageYrBlt      5.547945
GarageFinish     5.547945
GarageQual       5.547945
GarageCond       5.547945
BsmtFinType2     2.602740
BsmtExposure     2.602740
BsmtFinType1     2.534247
BsmtCond         2.534247
BsmtQual         2.534247
MasVnrArea       0.547945
Electrical       0.068493
dtype: float64
```

Changes in Target over Time

We've established that average price doesn't change significantly year over year (dataset covers the spun of 5 years, 2006-2010). However, in some housing categories we can see clear seasonality with the majority of sales happening during the summer months.







Baseline Model

As a baseline, we've built a simple Linear Regression model.

To do so we've selected features with the highest absolute value of the correlation coefficient, handled missing values among them and scaled values. We've eliminated features that highly correlate with each other in order to avoid multicollinearity.

```
MSE for OLS: 0.0320
RMSE for OLS: 0.1788
R^2 score for OLS: 0.8027
```

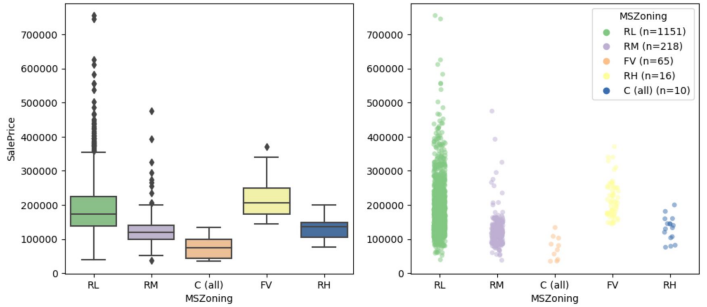
Further EDA and Feature Engineering

1. **Feature engineering**

After going through the entire dataset and checking the target breakdown by each feature, we've decided to try and create new features that accumulate the available data (e.g. total property area, including living area + lot area / total number of bathrooms in the house, including both basement and upper levels).

2. **Feature selection**

We've run a LGBMRegressor model on the dataset and selected top-80 features based on their importance.



Transformation Pipeline

Eventually, the dataset went through a multi-stage transformer, which included:

1. Filling missing values and removing outliers
2. Feature engineering
3. Normalization using PowerTransformer
4. Transformation of ordinal categorical features with OrdinalEncoder
5. Transforming categorical values with One-Hot Encoding
6. Scaling with StandardScaler

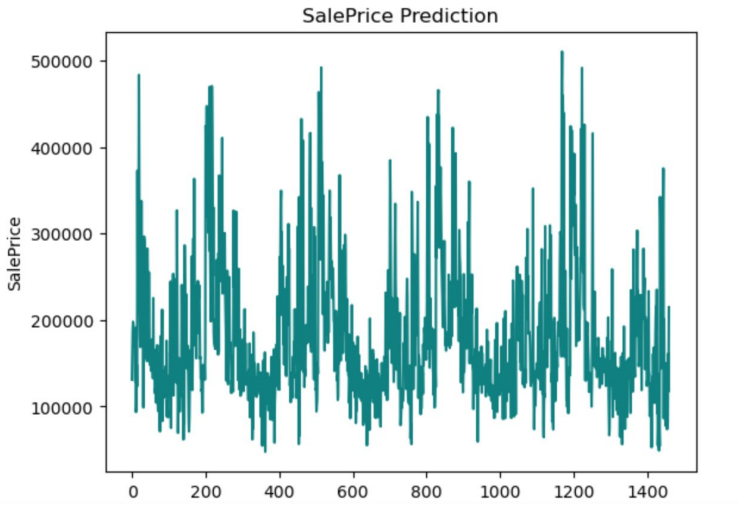
Applying Multiple Algorithms

Our next stage was to apply several regression algorithms to the dataset and compare their results on two estimators: RMSE and R2.

Linear models	Nonlinear models
Ridge	LGBMRegressor
Lasso	CatBoostRegressor
BayesianRidge	

Stacked Model

Next stage was to build a stacked model that would combine prediction of all algorithms. We used BayesianRidge for that. The model received predictions from each of the algorithms (Ridge, Lasso, LightGBM, CatBoost etc.) as an input and generated a unified prediction as an output.




```
# Stack

model = BayesianRidge()

model.fit(models_train_pred, y_test)

fin_pred = model.predict(models_test_pred)

# create submission file
subm = pd.DataFrame()
subm['Id'] = t_id.astype('int')
subm['SalePrice'] = np.expm1(fin_pred)
subm.set_index('Id').to_csv('submission.csv')
```

Optimizing Hyperparameters

Finally, we've used GridSearchCV to tune the hyperparameters for each part of the stacked model, which has lead to a slight improvement of the scores (RMSE and R2) on the 25% validation set.

	Ridge	BayesianRidge	Lasso	LGBMRegressor	CatBoostRegressor	mean
RMSE	0.127007	0.127160	0.125696	0.126913	0.119618	0.125279
R2	0.895781	0.895531	0.897922	0.895935	0.907555	0.898545