House prices

Goal: explore and model house prices (e.g., the Ames dataset with 79 features) to understand key price drivers and implement a baseline regression.
Methods: run EDA for correlations/outliers, encode features, and try linear or ridge regression with RMSE/RMSLE as evaluation, referencing competition conventions.
Deliverables: a preprocessing pipeline, feature importance/coefficients, and plots comparing predicted vs. actual sale prices.

That summary describes the **House Prices Prediction Project** perfectly. Here's a detailed, beginner-friendly expansion of that structure so you can use it as your **project plan and report outline** for your Ames dataset regression project (using only NumPy, pandas, and matplotlib—no scikit-learn).

Project Title

Predicting House Prices for Ames, Iowa

Goal

Explore and model house prices using the **Ames Housing dataset** (79 features) to identify the most influential property traits driving sale price and to **build a baseline regression model** (linear and ridge) evaluated via **RMSE or RMSLE** on log-transformed prices.[1] [2]

Objectives

- Perform **Exploratory Data Analysis (EDA)** to discover feature–price relationships and handle missing values.[3] [4]
- **Encode numerical and categorical data** manually with NumPy and pandas, using one-hot encoding written from scratch.[5] [3]
- Implement **Linear Regression and Ridge Regression** with **gradient descent** without using libraries like scikit-learn.[6] [7]
- Evaluate the model with **Root Mean Squared Logarithmic Error (RMSLE)** for realistic percentage-based accuracy.[8] [1]
- Visualize feature importance and predicted vs actual sale prices for interpretability.[4] [6]

Methods (Step-by-Step)

1. Data Loading

- Use pandas to read `train.csv` (with SalePrice) and `test.csv` (without SalePrice) from Kaggle's House Prices dataset. [2]
- Identify target variable `SalePrice` and separate numeric vs categorical features.

2. Exploratory Data Analysis (EDA)

- Plot **distribution of SalePrice** and of `log(SalePrice)` (log makes it more normally distributed, helpful for regression and RMSLE). [9] [10]
- Examine **missing values**, outliers (e.g., large GrLivArea with low SalePrice), and correlations between numeric features and price. [4] [6]

3. Preprocessing

- **Impute missing values**: numeric → median; categorical → mode or "None".
- **One-hot encode** categorical variables manually: create 0/1 indicator columns for each category in pandas. [11] [5]
- Normalize numeric columns by subtracting mean and dividing by standard deviation to help gradient descent converge. [7] [12]

4. Model Implementation

Build **Linear Regression from scratch** in NumPy:

- Prediction: $ \hat{y} = Xw + b $
- Loss (in log-scale): $ J(w,b) = \frac{1}{n} \sum (\hat{y} - y)^2 $
- Gradient updates:

$$w = w - \alpha \frac{\partial J}{\partial w}, \quad b = b - \alpha \frac{\partial J}{\partial b}$$

- Implement using a loop over a fixed number of iterations, tracking loss every 100 steps.

For **Ridge Regression**, modify weight updates to include the L2 penalty:

$$J(w,b) = \frac{1}{n} \sum (\hat{y} - y)^2 + \lambda ||w||^2$$

and update gradients accordingly:

$$dw = dw + \frac{2\lambda w}{n}$$

. [6] [7]

5. Evaluation

Compute **RMSLE**:

$$\mathrm{RMSLE} = \sqrt{\frac{1}{n}\sum(\log(\hat{y}_i + 1) - \log(y_i + 1))^2}$$

—Since you trained on log(SalePrice), RMSE on log-scale equals RMSLE on raw prices.[1] [8]

Plot:

- **Actual vs Predicted SalePrice** (scatter plot around the diagonal line).
- **Residuals (errors)** distribution to see bias in predictions.

6. Deliverables

1. **A Python notebook/report** (`.ipynb` or `.py`) with:
   - Step-by-step EDA
   - Feature engineering and encoding
   - Linear regression training (with gradient descent)
   - RMSLE score reporting
   - Ridge regression comparison
2. **Visualizations** (matplotlib):
   - Correlation heatmap
   - Predicted vs Actual SalePrice scatter plot
   - Residual plots
3. **Discussion**:
   - Top correlated features with SalePrice (e.g., OverallQual, GrLivArea, GarageCars).
   - How log-transform and one-hot encoding improved results.
   - Difficulties or errors fixed manually.

Suggested Timeline (for a 2-person team)

Phase	Main Tasks	Lead	Time Estimate
1	Data loading, EDA plots	Member 1	1–2 days
2	Preprocessing & encoding	Member 1	2–3 days
3	Linear regression from scratch	Member 2	3–4 days
4	RMSLE evaluation, ridge regression, plots	Both	2–3 days
5	Report writing & code cleanup	Both	1–2 days

Dataset Download Links

1. **Ames House Prices dataset** (official Kaggle competition):
   https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data
   - train.csv (with SalePrice)
   - test.csv (without SalePrice)
   - data_description.txt (feature dictionary)
   - sample_submission.csv (reference format)
2. **Alternative simple dataset (no login required)**:
   https://www.kaggle.com/datasets/yasserh/housing-prices-dataset

Final Deliverables Summary

- **Notebook/report** fully explaining math + code
- **Custom linear regression functions in NumPy**
- **Plots**: correlation, actual vs predicted, residuals
- **RMSLE metric results** (with discussion of tuning and regularization)

Would you like a **ready-to-follow file & folder structure (with starter code templates)** for this project as your next step? That will make it easy for you and your partner to start coding immediately.

⁂

1. https://www.kaggle.com/competitions/home-data-for-ml-course
2. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data
3. https://jse.amstat.org/v19n3/decock.pdf
4. https://www.kaggle.com/code/siddheshpujari/eda-and-prediction-of-house-price
5. https://www.machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/
6. https://www.kaggle.com/code/ryanholbrook/feature-engineering-for-house-prices
7. https://realpython.com/gradient-descent-algorithm-python/
8. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/discussion/75817
9. http://d2l.ai/chapter_multilayer-perceptrons/kaggle-house-price.html
10. https://www.linkedin.com/pulse/understanding-machine-learning-model-evaluation-metrics-ayush-gupta-ixj5e
11. https://stackoverflow.com/questions/37292872/how-can-i-one-hot-encode-in-python
12. https://www.kdnuggets.com/linear-regression-from-scratch-with-numpy