Bulldozer Price Prediction Project — Full Explanation

This document explains the steps I followed to complete the **Bluebook for Bulldozers Machine Learning project**. It describes why each step was used, how it works, and what I learned.

1. Problem Definition

The goal of this project is to predict the auction sale price of bulldozers. This is a regression problem since the target variable (SalePrice) is continuous.

2. Data Loading & Parsing

I loaded the dataset with pandas and used **parse_dates=['saledate']**. Parsing dates allows easy extraction of features like year, month, day, day of week, and day of year.

3. Feature Engineering

From *saledate*, I created new columns: saleYear, saleMonth, saleDay, saleDayOfWeek, saleDayOfYear. These features help the model learn seasonal patterns and auction timing effects. I dropped the original *saledate* to avoid data leakage.

4. Handling Missing Values

For numeric features: I filled missing values with the **median**. Median is robust to outliers and avoids bias from extreme values. I also added a binary *_is_missing* column to indicate where values were missing. For categorical features: I converted them to **pandas.Categorical codes** and added missing indicators. This ensures the model knows when data was missing.

5. Converting Categorical Variables

I used **pd.Categorical(...).codes + 1** to transform string categories into numeric codes. This is important because scikit-learn models can only work with numbers. Adding +1 ensures missing values are not confused with a valid category code.

6. Splitting Data

I used a **time-based split**: training data is from earlier years, validation is from later years. This simulates real-world forecasting, since we predict future sales from past sales.

7. Model Choice

I chose **RandomForestRegressor** because: - It handles missing values and categorical codes well. - It is robust to overfitting. - It provides **feature importances** to interpret the model. I tuned hyperparameters: n_estimators, max_features, min_samples_split, max_samples, n_jobs.

8. Evaluation Metrics

I used three metrics: - **MAE (Mean Absolute Error)**: average absolute difference between predictions and actual prices. - **RMSLE (Root Mean Squared Log Error)**: Kaggle competition metric, compares relative errors. - **R^2 (Coefficient of Determination)**: measures how much variance in target is explained by the model. These metrics together give a full picture of performance.

9. Feature Importance

I extracted and plotted **feature_importances_** from the Random Forest. This showed which features had the biggest impact on price prediction, e.g., YearMade, saleYear, ProductSize.

10. Test Predictions & Submission

Before predicting on test data, I aligned columns with training using: *test_df = test_df.reindex(columns=x_train.columns, fill_value=0)*. Then I predicted and created a CSV with **SalesID** and **SalePrice** for Kaggle submission.

11. Results

Final scores: - Training $R^2 \approx 0.86$, Validation $R^2 \approx 0.83$ - Training MAE ≈ 5561, Validation MAE ≈ 7177 - Training RMSLE ≈ 0.25, Validation RMSLE ≈ 0.29 This shows the model generalizes well with slight gap between train and validation.

12. Learnings

From this project, I learned: - How to engineer date features from timestamps. - Handling missing data with median + missing indicators. - Encoding categorical features. - Avoiding data leakage with proper splits. - Evaluating regression with MAE, RMSLE, R^2. - Training and tuning Random Forest for tabular prediction. - Preparing predictions for Kaggle submission.