Auto-MPG Prediction: A Regression Analysis

This project explores the "Auto-MPG" dataset from Kaggle to predict a car's fuel efficiency (miles per gallon, or MPG). The analysis begins with data cleaning and exploratory data analysis (EDA), establishes a baseline linear regression model, and then improves upon it using polynomial regression and manual feature engineering.

The final model is built using a clean `scikit-learn` `Pipeline` and `ColumnTransformer` to ensure reproducible results and prevent data leakage.

Dataset

- **Source:** Kaggle: Auto-MPG Dataset (Originally from the UCI ML Repository)
- **Instances:** 398
- **Target Variable:** `mpg`
- **Features:**
    - `cylinders`: (Categorical)
    - `displacement`: (Continuous)
    - `horsepower`: (Continuous)
    - `weight`: (Continuous)
    - `acceleration`: (Continuous)
    - `model year`: (Categorical/Temporal)
    - `origin`: (Categorical: 1: USA, 2: Europe, 3: Japan)
    - `car name`: (String - Dropped for modeling)

Project Workflow

The project followed a systematic approach to model development and refinement.

1. Data Cleaning

- **Missing Values:** The `horsepower` column contained 6 missing values (represented as '?'). These were converted to numeric and imputed using the mean `horsepower`.
- **Categorical Features:** The `origin` column, though numeric, represents categories. It was one-hot encoded into `origin_1`, `origin_2`, and `origin_3`.

2. Exploratory Data Analysis (EDA)

- **Correlations:** A heatmap and scatter plots revealed strong negative correlations between `mpg` and features like `weight`, `horsepower`, and `displacement`. This makes sense, as heavier, more powerful cars are typically less fuel-efficient.
- **Temporal Trend:** A bar plot of `model year` vs. `mpg` showed a clear trend of increasing fuel efficiency over time, with a notable jump in the mid-70s, likely due to the 1973 oil crisis.

3. Model 1: Baseline Linear Regression

A simple linear regression model was trained on the cleaned and scaled features.

- **Result (10-Fold CV): $R^2 \approx 0.816$**
- This baseline was good, but the "Actual vs. Predicted" plot and residual plots from the notebook indicated non-linear patterns that the model was not capturing.

4. Model 2: Polynomial Regression

To capture the observed non-linear relationships (e.g., the curve in the `weight` vs. `mpg` plot), a `Pipeline` containing `PolynomialFeatures` was used. We looped through different degrees to find the best fit.

- **Finding the Best Degree:** A plot of the polynomial degree vs. the cross-validated Mean Squared Error (MSE) showed that `degree=2` provided the best fit. Degree 1 (linear) was too simple (high bias), while Degree 3+ overfit the data and led to a massive increase in error (high variance).
- **Result (10-Fold CV, Degree 2): $R^2 \approx 0.864$**
- This confirmed that a quadratic model significantly outperformed the simple linear model.

5. Model 3: Feature Engineering

Instead of relying on `PolynomialFeatures` to find interactions, we manually created new features based on our domain knowledge and EDA findings.

- **Interaction/Ratio Features:**
  - `power_to_weight` = `horsepower` / `weight`
  - `weight_x_accel` = `weight` * `acceleration`
  - `disp_per_cyl` = `displacement` / `cylinders`
- **Categorical Binning:**
  - To better capture the "jump" in efficiency in the 70s, `model year` was binned into eras (e.g., '70-73', '74-76', etc.) and then one-hot encoded. This treats the eras as distinct categories rather than a continuous number.

6. Model 4: Final Pipeline (Engineered Features)

A final `Pipeline` was constructed using a `ColumnTransformer`. This pipeline correctly applies `StandardScaler` *only* to the new list of continuous features while passing through the (now expanded) list of one-hot encoded features.

- **Result (10-Fold CV): $R^2 \approx 0.864$**
- This result is a key insight: our manually engineered features performed just as well as the 2nd-degree polynomial model. This validates our hypotheses from the EDA.

Model Performance Summary

Model Description	Features Used	10-Fold Cross-Validation R^2
Model 1: Baseline	Standard Linear Regression (Scaled)	0.816
Model 2: Polynomial	Polynomial Regression (Degree 2)	**0.864**
Model 4: Final	Linear Regression (Engineered Features)	**0.864**

Final Model Visualization

The "Actual vs. Predicted" plot for the final model (Model 4) shows a much tighter cluster around the perfect-prediction line compared to the baseline, confirming its improved accuracy.

How to Run

1. **Libraries:** This project requires `pandas`, `numpy`, `matplotlib`, `seaborn`, and `scikit-learn`.
2. **Data:** The `auto-mpg.csv` file (not included in this repository) must be in the same directory as the notebook.
3. **Execution:** The complete analysis is contained within the `Auto-mpg dataset kaggle.ipynb` notebook. Cells can be run in order to reproduce the findings.

Conclusion & Next Steps

The analysis demonstrates that a simple linear regression model ($R^2 \approx 0.82$) is insufficient for this dataset. The relationship between a car's attributes and its MPG is non-linear.

By capturing these 2nd-degree relationships—either through `PolynomialFeatures` or (as shown in the final model) through thoughtful manual feature engineering—we can significantly improve the model's performance to a reliable **R^2 of ~0.86**.

A potential next step would be to try non-linear models, such as a **Random Forest Regressor** or **XGBoost Regressor**, which might be able to capture even more complex interactions and further improve the predictive score.