

USED CAR PRICE PREDICTION SYSTEM

Web Scraping eBay to Deploying Predictive Model

Mujahid Afzal



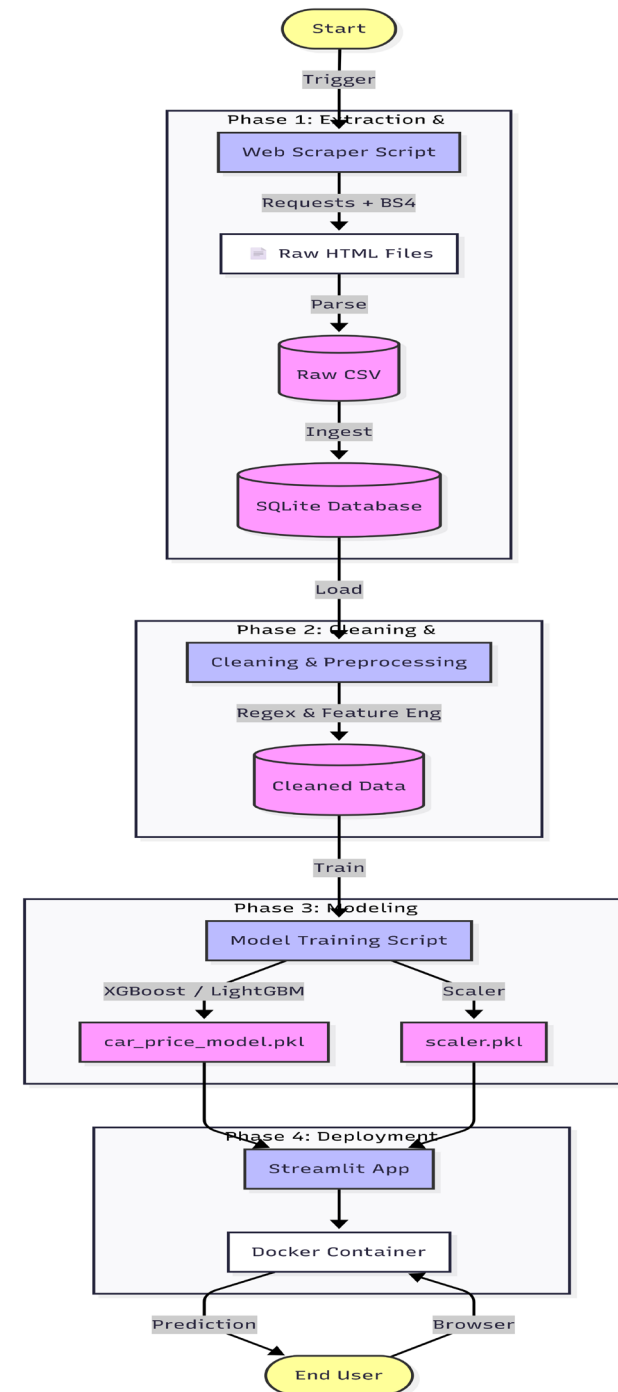
PROJECT OVERVIEW & OBJECTIVE

- Goal: Build a Machine Learning model to predict fair market value for used cars using real-time data.
- Why eBay?
 - - Represents the true 'open market' with high volume.
 - - Data is unstructured (messy titles), offering a real Data Science challenge.
- Tech Stack: Python, BeautifulSoup, SQLite, Pandas, XGBoost, Streamlit, Docker.



THE ARCHITECTURE (PIPELINE)

- End-to-End Workflow:
- 1. Web Scraping: Custom script to download thousands of listings.
- 2. Storage: Raw data stored in SQLite Database for persistence.
- 3. Cleaning: Regex & Feature Engineering to fix messy text.
- 4. Modeling: Benchmarking 8 models (Linear vs Tree-based).
- 5. Deployment: Dockerized Streamlit App.

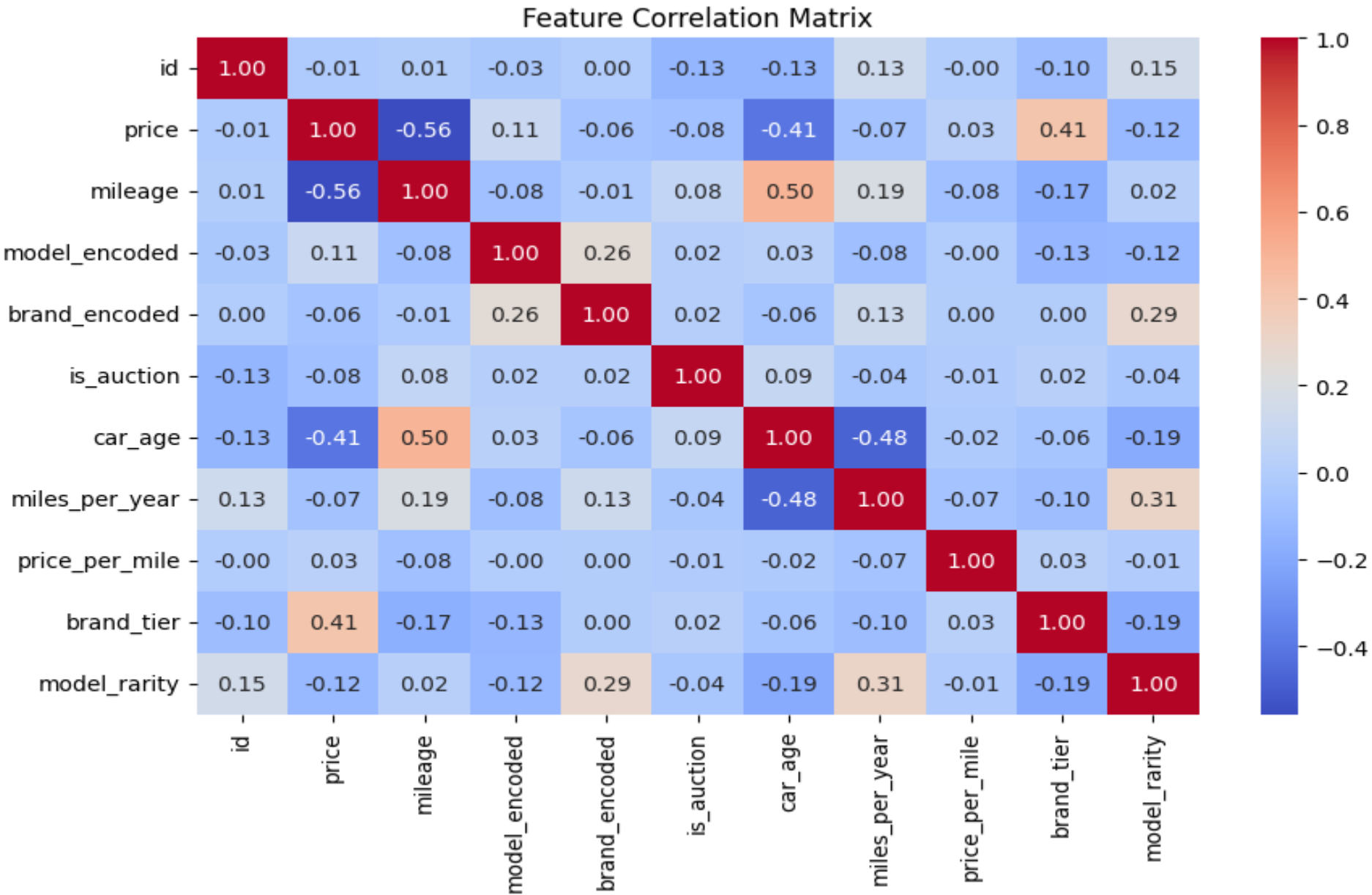


DATA CLEANING

- Challenge: Raw data was unusable (e.g., '🔥 2015 Toyota Camry!!! L@@K 74k').
- Solutions:
 - - Regex: Extracted sticky years (e.g., 'listing2016').
 - - Logic: Normalized mileage ('74k' -> 74,000).
 - - 'Ghost Car' Fix: Injected 'Year + Brand' if cleaning wiped the title.
- Result: 100% of valid rows preserved for training.



FEATURE CORRELATION MATRIX

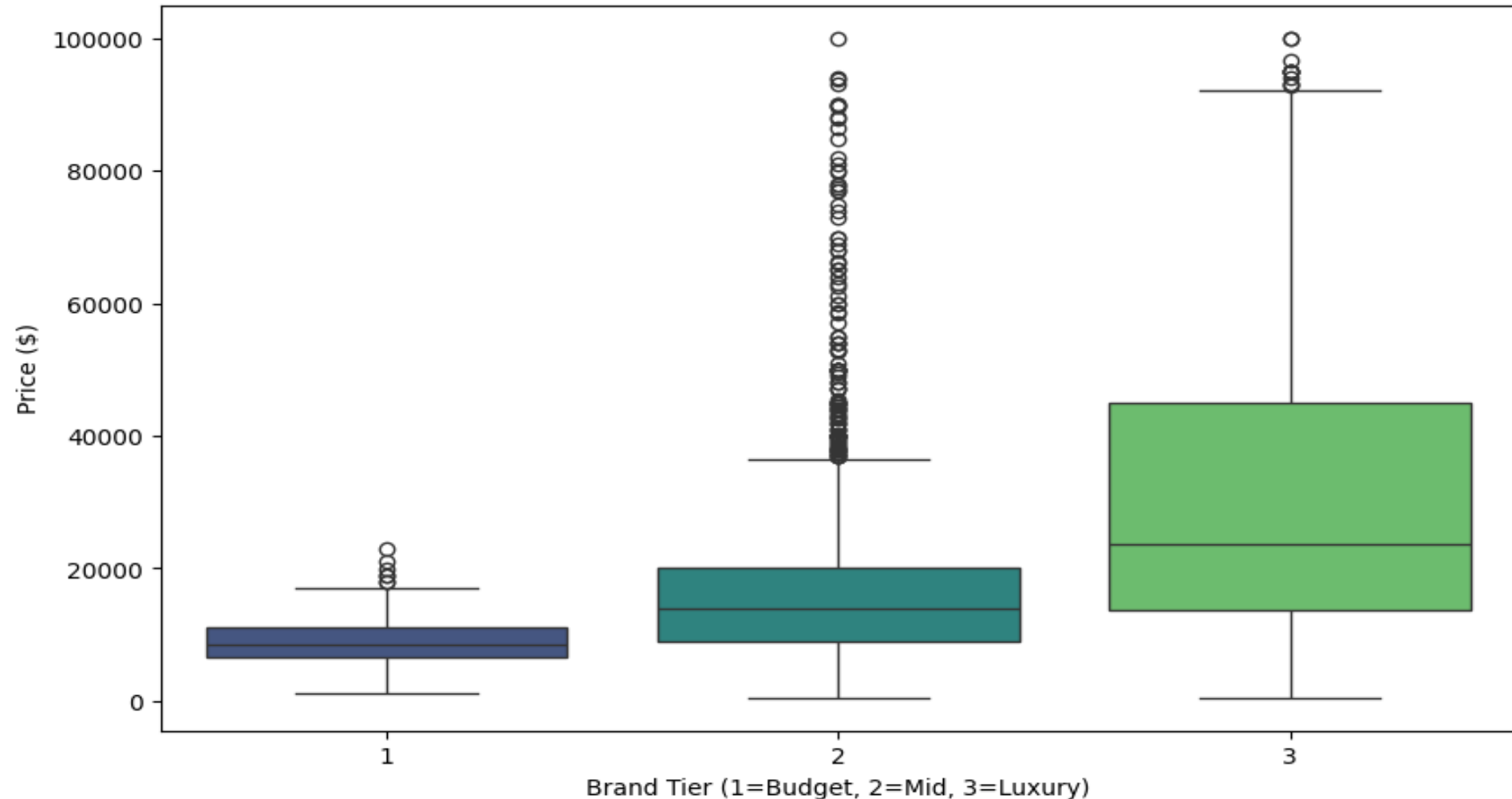


FEATURE ENGINEERING

■ Brand Tier:

- Mapped brands to distinct categories (1=Economy, 2=Mid-Range, 3=Luxury).
- Captured the "Luxury Premium," proving a +0.41 correlation with price.

Price Distribution by Brand Tier



FEATURE ENGINEERING

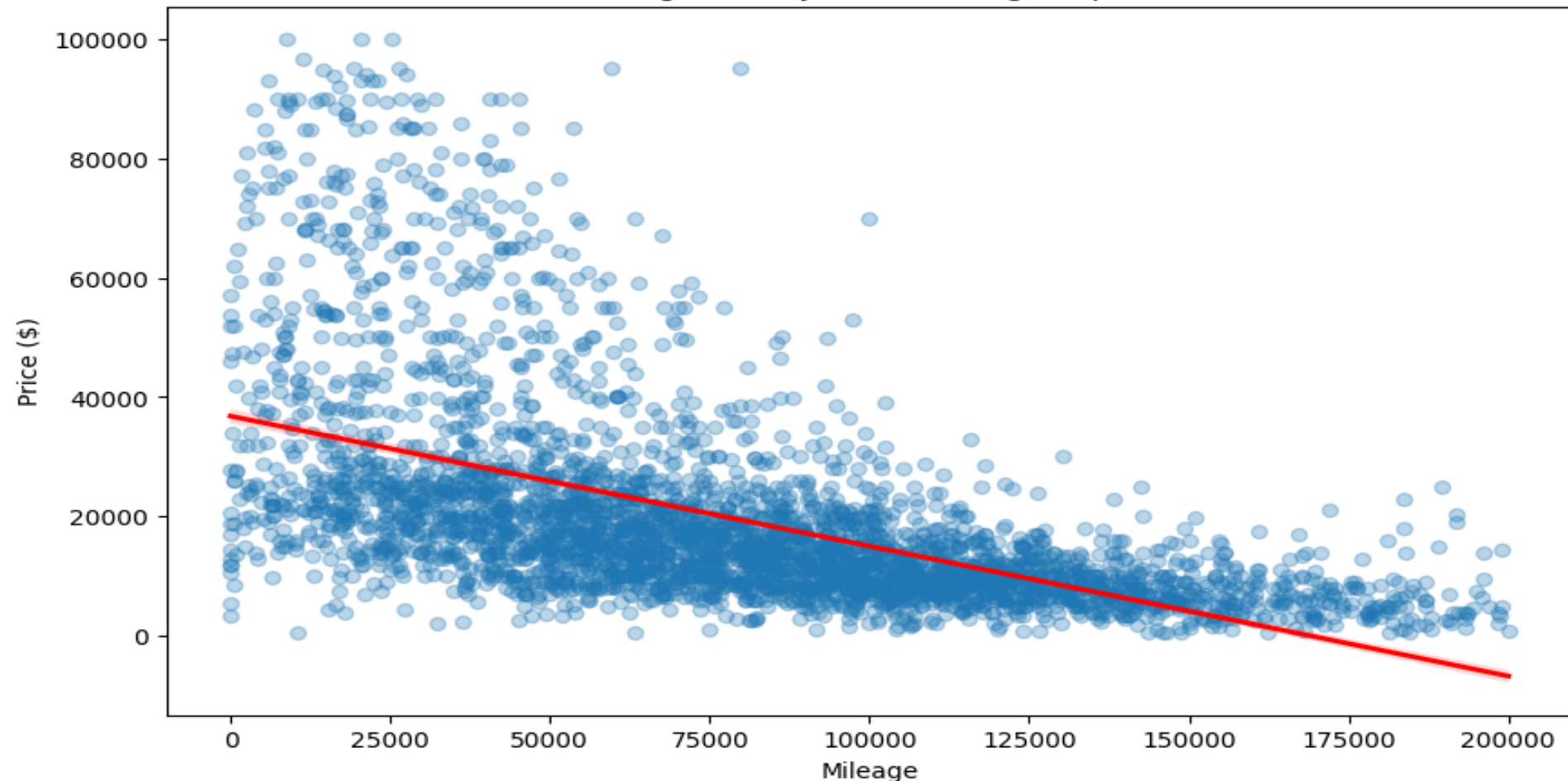
- The Depreciation Curve (Justifies Price vs. Age: -0.41)
 - This visualization validates the impact of car_age on value. It reveals the non-linear nature of depreciation.
 - A steep drop in value during the first few years (0–5), followed by a flattening of the curve as cars get older.



FEATURE ENGINEERING

- The "Mileage Penalty" (Justifies Price vs. Mileage: -0.56)
 - This chart visually proves that as mileage increases, the price drops reliably. The red regression line helps confirm this negative trend, validating why mileage is likely strongest predictor.

The "Mileage Penalty": How Mileage Impact Price



ANALYSIS & MODEL RESULTS

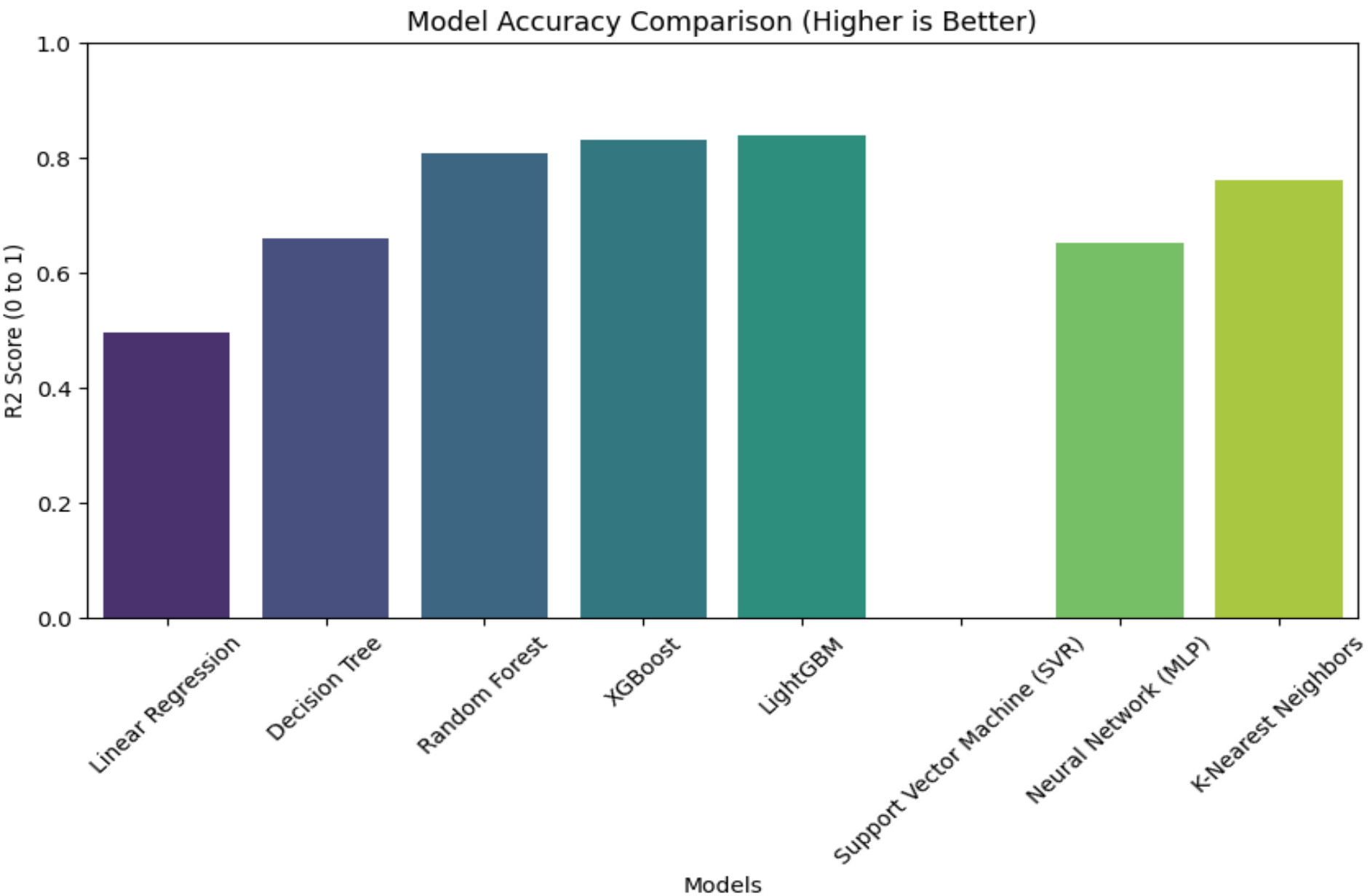
- Insights:
- Car Age: Calculated `2025 - Year` to measure depreciation directly.
- Mileage is also a strong feature followed by model.
- - Seller Score proved to be noise (0% correlation) that's why it was excluded.
- Winning Model: XGBoost
- - **Accuracy:** ~84.5% (R2 Score)
- **MAE:** ~\$4,300 (Average Error)
- **Speed:** Training time 4.3s (vs LightGBM 53s). XGBoost was 10x faster in your specific tuning run.
- - Outperformed Linear Regression (~50%) significantly.



MODEL ACCURACY COMPARISON

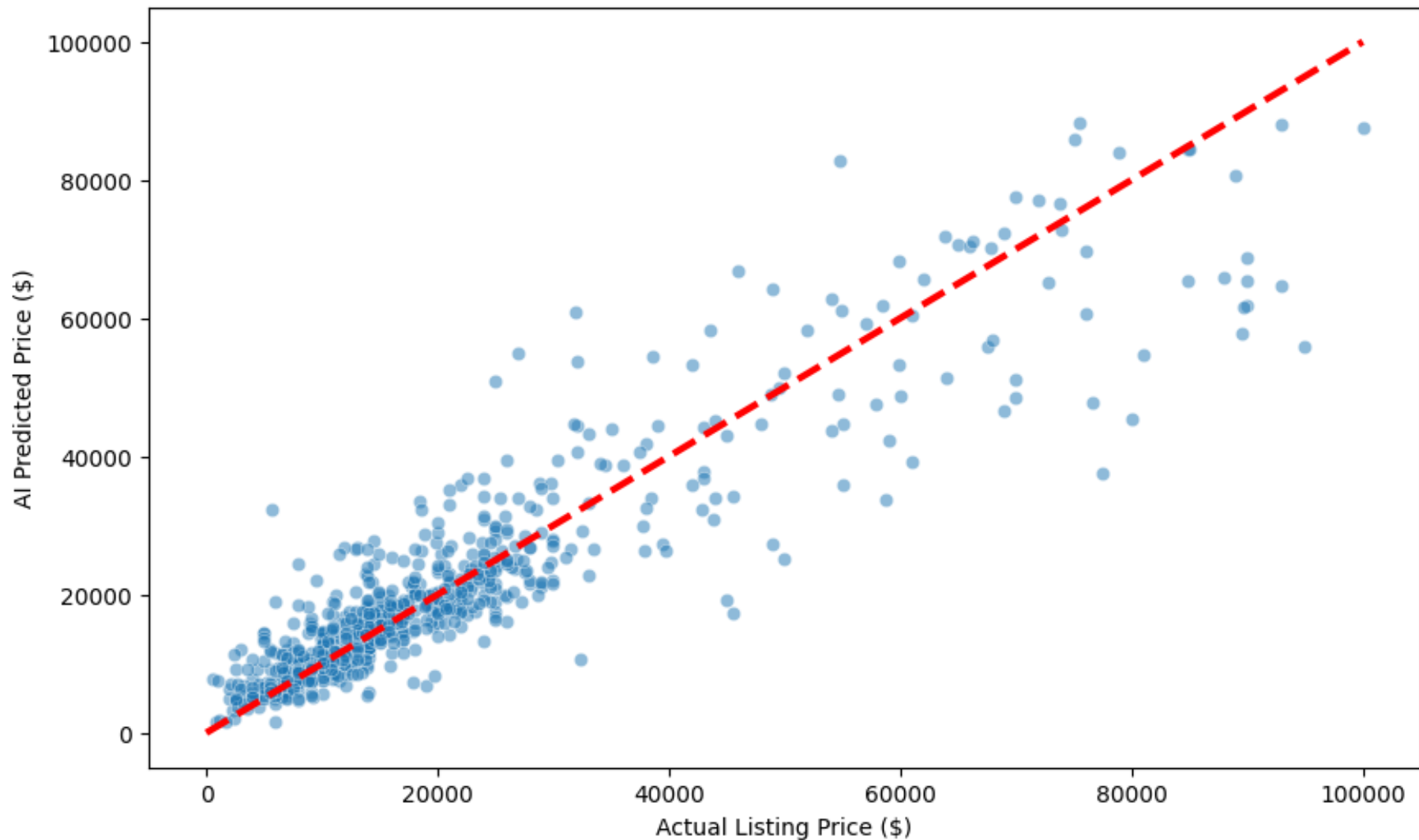
Rank	Model	R ² Score	MAE (Error)	Time (s)	Output
1	XGBoost	84.49%	~\$4,299	3.1s	The Winner. Highest accuracy (\$R^2\$), lowest error, and fastest training speed.
2	LightGBM	84.39%	~\$4,378	51.2s	Runner Up. Comparable accuracy to XGBoost but significantly slower training time.
3	Random Forest	81.11%	~\$4,811	11.2s	Solid performance (>80%), but higher error than the gradient boosting models.
4	KNN	76.14%	~\$5,223	-	Effective baseline; confirms that "similar cars have similar prices."
5	Decision Tree	65.90%	~\$6,417	-	Prone to overfitting; struggled to generalize to unseen data.
6	Neural Network	65.21%	~\$6,693	-	Underperformed tree-based models on this specific tabular dataset.
7	Linear Regression	49.55%	~\$8,573	-	Poor fit. Proves car pricing is highly non-linear.
8	SVM (SVR)	-7.7%	~\$11,041	-	Failed model. Could not effectively separate the data hyperplanes.

MODEL ACCURACY COMPARISON



ACTUAL PRICE VS PREDICTED PRICE

Actual Price vs. Predicted Price



DEPLOYMENT & CONCLUSION

Final Product: A Dockerized Streamlit App.

- - Users select a brand and get instant price estimates.
- - 'Scrap Guard' logic prevents negative price predictions.

Conclusion:

- Architected a complete lifecycle: **Scraping** → **Warehousing** → **Feature Engineering** → **Deployment**
- Delivered High-Precision AI: Engineered a model achieving 84.5% Accuracy, outperforming traditional regression baselines by over 35%.
- Optimized for Production: Selected XGBoost for its superior efficiency, reducing training time to 3.1s while maintaining a low error margin (~\$4,300 MAE).
- Scalable Architecture: Built a robust pipeline capable of handling thousands of new listings without manual intervention.

