Second-Hand Vehicle Price Prediction Using a Medallion Architecture Pipeline on Databricks

Group 8: Mark Racca, Ranjit Singh, Edmund Yu, ENSF 612

***Abstract*—This paper presents a comprehensive machine learning pipeline for predicting second-hand vehicle prices using a medallion architecture implemented on Databricks. We processed approximately 24,000 vehicle listings from the Toronto area through a three-layer data architecture (Bronze, Silver, Gold) using PySpark and Delta Lake, reducing the dataset to 19,461 high-quality records after rigorous cleaning and validation. Three regression models were evaluated: Linear Regression (baseline), Random Forest, and Gradient Boosting. The Gradient Boosting model achieved the best performance with an RMSE of $16,399 and an R^2 score of 0.9090, demonstrating strong predictive capability for vehicle pricing. Linear Regression provided a reasonable baseline (R^2 = 0.7034), while Random Forest achieved intermediate performance (R^2 = 0.8504). Our hybrid architecture, combining PySpark for data engineering with scikit-learn for machine learning, successfully addresses the constraints of Databricks Community Edition while maintaining production-quality data processing patterns. This work demonstrates the practical application of big data engineering principles to real-world price prediction problems.**

***Index Terms*—Machine Learning, Vehicle Price Prediction, Medallion Architecture, Delta Lake, PySpark, Gradient Boosting, Random Forest, Databricks.**

I. Introduction

THE used vehicle market represents a significant segment of the automotive industry, with millions of transactions occurring annually. Accurate price prediction for second-hand vehicles is crucial for both buyers and sellers, enabling fair market valuations and informed decision-making. However, vehicle pricing is inherently complex, influenced by numerous factors including make, model, age, mileage, condition, and regional market dynamics.

Traditional pricing approaches often rely on manual appraisals or simple rule-based systems that fail to capture the intricate relationships between vehicle attributes and market value. Machine learning offers a data-driven alternative, capable of learning complex patterns from historical transaction data to generate accurate price predictions.

This paper presents a complete end-to-end machine learning pipeline for vehicle price prediction, implementing industry best practices for data engineering and model development. Our key contributions include:

- Implementation of a medallion architecture (Bronze $\rightarrow$ Silver $\rightarrow$ Gold) for structured data processing using Delta Lake on Databricks
- Comprehensive data cleaning and feature engineering pipeline processing 24,198 raw vehicle records
- Comparative evaluation of three regression models with hyperparameter optimization
- A hybrid PySpark/scikit-learn architecture that addresses Databricks Community Edition constraints while maintaining production patterns

The remainder of this paper is organized as follows: Section II describes the dataset and exploratory analysis. Section III details our methodology, including the medallion architecture and feature engineering approaches. Section IV presents experimental results and model comparisons. Section V discusses findings, limitations, and potential improvements. Section VI concludes with key insights and future work directions.

This paper was produced as part of the ENSF 612 course project at the University of Calgary, December 2025.
Dataset source: Kaggle - Used Vehicles for Sale [1].

II. Dataset Description

A. Data Source

The dataset comprises 24,198 second-hand vehicle listings collected from the Toronto area, sourced from Kaggle [1]. Each record represents a unique vehicle listing with 17 attributes capturing vehicle specifications and pricing information.

B. Feature Overview

The raw dataset contains the following attributes:

Identification Features: Make, Model, Year

Physical Specifications: Body Type, Engine, Transmission, Drivetrain, Exterior Colour, Interior Colour, Passengers, Doors

Usage Metrics: Kilometres (odometer reading)

Fuel Efficiency: City (L/100km), Highway (L/100km)

Fuel Type: Gasoline, Diesel, Electric, Hybrid, Premium Unleaded, etc.

Target Variable: Price (CAD)

C. Initial Data Quality Assessment

Exploratory analysis revealed significant data quality challenges requiring systematic cleaning. Table I summarizes the missing value distribution in the raw Bronze layer data.

The Passengers field exhibited the highest missing rate at 50.31%, followed by Interior Colour at 32.15%. Critical prediction features (Year, Make, Model, Price) contained no missing values, providing a solid foundation for modeling.

TABLE I
MISSING VALUE ANALYSIS IN BRONZE LAYER

Column	Null Count	Null %
Passengers	12,173	50.31%
Interior Colour	7,780	32.15%
City (Fuel)	6,363	26.30%
Highway (Fuel)	6,363	26.30%
Doors	4,587	18.96%
Engine	2,062	8.52%
Transmission	1,344	5.55%
Drivetrain	1,231	5.09%
Body Type	1,230	5.08%
Kilometres	233	0.96%
Year, Make, Model, Price	0	0.00%

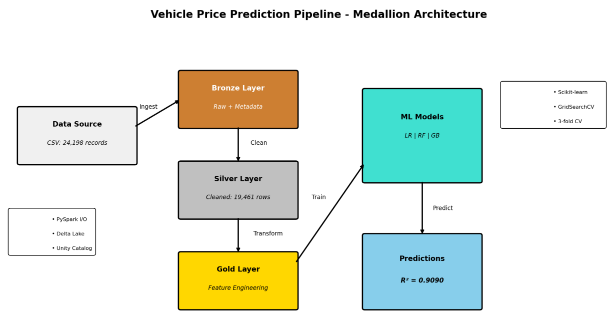

Fig. 1. Vehicle Price Prediction Pipeline implementing the Medallion Architecture pattern. Data flows from raw CSV through Bronze (raw ingestion), Silver (cleaning), and Gold (feature engineering) layers before model training and prediction.

D. Price Distribution

The target variable exhibits a right-skewed distribution with:

- Minimum: \$1,500
- Maximum: \$919,988
- Mean: \$47,569
- Median: \$34,988

The substantial difference between mean and median indicates the presence of high-value luxury vehicles that create positive skewness. The wide price range (\$1,500 to \$919,988) presents challenges for model training, particularly for linear models sensitive to outliers.

III. METHODOLOGY

A. System Architecture

We implemented a medallion architecture, a layered data design pattern that progressively refines data quality through distinct processing stages. This architecture was deployed on Databricks Community Edition using Delta Lake for data storage and versioning. Fig. 1 illustrates the complete pipeline architecture.

1) Bronze Layer - Raw Data Ingestion: The Bronze layer ingests raw CSV data with minimal transformation, preserving the original data while adding metadata for lineage tracking:

- `record_id`: Unique identifier using monotonically increasing ID
- `ingestion_timestamp`: UTC timestamp of data ingestion
- `source_file`: Original filename for provenance tracking

Column names were standardized by removing leading/trailing spaces and replacing spaces with underscores to comply with Delta Lake naming requirements. The Bronze table retains all 24,198 records in their original form.

2) Silver Layer - Data Cleaning and Validation: The Silver layer applies comprehensive data cleaning and validation rules. Table II summarizes the key transformations applied.

TABLE II
SILVER LAYER DATA CLEANING OPERATIONS

Operation	Description
Kilometres Extraction	Regex extraction from “53052 km” format to numeric
Fuel Efficiency	Extract numeric L/100km values; handle range formats (e.g., “9.0L - 9.5L/100km” → 9.0)
Categorical Standardization	Normalize fuel types (“Gas” → “Gasoline”), uppercase transmission values
Missing Value Imputation	Group-wise median imputation for Kilometres by Make/Model/Year
Outlier Removal	Filter: Year ∈ [1990, 2025], Price ∈ [\$1,000, \$1,000,000], Kilometres ∈ [0, 500,000]
Duplicate Removal	Remove exact duplicates based on all feature columns

The cleaning process removed 127 outlier records and 4,610 duplicate entries, reducing the dataset from 24,198 to 19,461 records (19.58% reduction). This aggressive cleaning ensures high data quality for downstream modeling while retaining sufficient samples for robust training.

3) Gold Layer - Feature Engineering: The Gold layer transforms cleaned data into model-ready features through several engineering operations:

Derived Numerical Features:

- `vehicle_age`: Current year minus vehicle year
- `avg_fuel_efficiency`: Mean of city and highway fuel consumption
- `engine_displacement`: Extracted from engine string (e.g., “2.0L” → 2.0)
- `cylinder_count`: Extracted from engine description (e.g., “V6” → 6)

Frequency Encoding: Due to the high cardinality of the Model feature (hundreds of unique values), we applied frequency encoding with log transformation:

$$\texttt{model\_frequency\_log} = \ln(\text{count}(\text{Model}) + 1) \quad (1)$$

This approach captures the intuition that popular models may have more stable market pricing while avoiding the dimensionality explosion of one-hot encoding.

Train/Test Split: The Gold layer data was split into training (80%, 15,634 records) and test (20%, 3,827 records) sets using a fixed random seed for reproducibility.

B. Hybrid Architecture Decision

A significant architectural decision was required due to Databricks Community Edition limitations. The free tier uses Spark Connect, which blocks direct instantiation of PySpark ML transformers (StringIndexer, OneHotEncoder) and models through its security manager.

We adopted a hybrid approach:

- **Data Layer**: PySpark for all data I/O and transformations
- **ML Layer**: scikit-learn for preprocessing and model training
- **Results Layer**: PySpark for persisting predictions to Delta tables

This architecture maintains production-quality data engineering patterns while circumventing platform restrictions. The trade-off is that model training occurs on data converted to Pandas DataFrames, limiting scalability to datasets that fit in memory.

C. Feature Preprocessing Pipeline

The scikit-learn preprocessing pipeline consists of:

Categorical Features (5 columns: Make, BodyType, FuelType, Transmission, Drivetrain):

- OneHotEncoder with `handle_unknown='ignore'` and `drop='first'` to handle unseen categories during inference and eliminate multicollinearity from the dummy variable trap

Numerical Features (8 columns: Kilometres, City, Highway, vehicle_age, avg_fuel_efficiency, engine_displacement, cylinder_count, model_frequency_log):

- StandardScaler for zero-mean, unit-variance normalization

D. Model Selection and Hyperparameter Tuning

Three regression algorithms were evaluated:

1) Linear Regression (Baseline): Ordinary least squares regression serving as a baseline model. No hyperparameter tuning was applied.

2) Random Forest Regressor: An ensemble method using bootstrap aggregation of decision trees. Hyperparameters were tuned using 3-fold cross-validation with GridSearchCV:

- `n_estimators`: [50, 100, 200]
- `max_depth`: [5, 10, 15]
- `min_samples_leaf`: [1, 5]

Best parameters: `n_estimators`=50, `max_depth`=10, `min_samples_leaf`=5

3) Gradient Boosting Regressor: A sequential ensemble method that builds trees to correct residual errors. Hyperparameters tuned:

- `n_estimators`: [50, 100, 200]
- `max_depth`: [3, 5, 7]
- `learning_rate`: [0.05, 0.1, 0.2]

Best parameters: `n_estimators`=50, `max_depth`=7, `learning_rate`=0.2

TABLE III
Model Performance Comparison on Test Set

Model	RMSE	MAE	R^2	Time (s)
Gradient Boosting	\$16,399	\$6,553	0.9090	187.7
Random Forest	\$21,023	\$8,864	0.8504	147.6
Linear Regression	\$29,605	\$13,592	0.7034	2.4

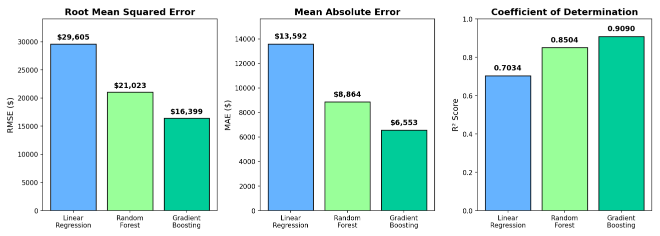

Fig. 2. Visual comparison of model performance metrics. Gradient Boosting achieves the lowest error and highest R^2, followed by Random Forest and Linear Regression.

E. Evaluation Metrics

Model performance was assessed using three standard regression metrics:

Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

Coefficient of Determination (R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

IV. Results

A. Model Comparison

Table III presents the performance comparison across all three models on the held-out test set.

The Gradient Boosting model achieved the best overall performance with an RMSE of \$16,399 and R^2 of 0.9090, indicating that the model explains approximately 91% of the variance in vehicle prices. The Random Forest model performed well with R^2 of 0.8504, while Linear Regression provided a reasonable baseline with R^2 of 0.7034, explaining approximately 70% of the price variance. The performance hierarchy aligns with expectations: ensemble methods outperform simple linear models on this non-linear pricing problem, while Linear Regression's lower performance reflects its inability to capture complex feature interactions without explicit polynomial terms.

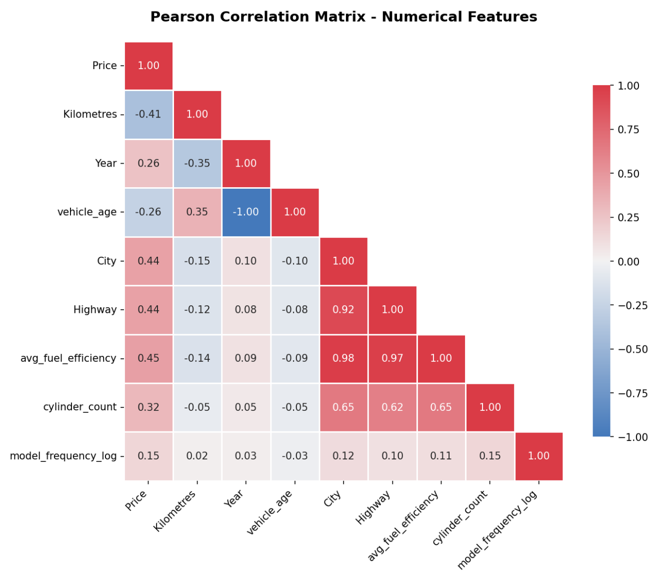

Fig. 3. Pearson correlation matrix for numerical features. Kilometres shows moderate negative correlation with price (-0.41), while average fuel efficiency shows moderate positive correlation (0.45).

B. Feature Importance Analysis

Fig. 3 shows the correlation matrix for numerical features, revealing key relationships with the target variable (Price).

Key correlation findings:

- `Kilometres` shows moderate negative correlation with Price (-0.41), confirming mileage as a key pricing factor
- `vehicle_age` shows negative correlation with Price (-0.26), confirming depreciation effects
- `avg_fuel_efficiency` shows positive correlation (0.45), indicating fuel efficiency impacts pricing
- City and Highway fuel efficiency show strong positive correlations with Price (0.44 and 0.44 respectively)

C. Prediction Quality

Fig. 4 presents the actual versus predicted price scatter plot for the Gradient Boosting model.

The model demonstrates strong predictive performance across the price range, with predictions clustering tightly around the perfect prediction line. Some dispersion is observed at higher price points ($100K+), likely due to the lower sample density in this range and the inherent variability in luxury vehicle pricing.

D. Price Distribution Analysis

Fig. 5 illustrates the price distribution characteristics of the dataset.

The analysis reveals that SUVs command the highest median prices, followed by Sedans and Trucks. This aligns with market trends favoring larger utility vehicles in the Canadian market.

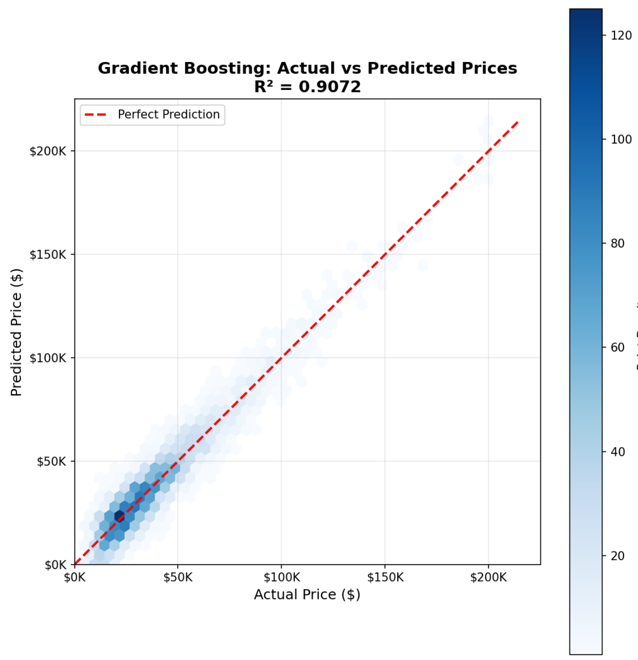

Fig. 4. Actual vs. Predicted prices for the Gradient Boosting model (R^2 = 0.9090). Points clustered along the diagonal indicate accurate predictions. Color intensity represents point density.

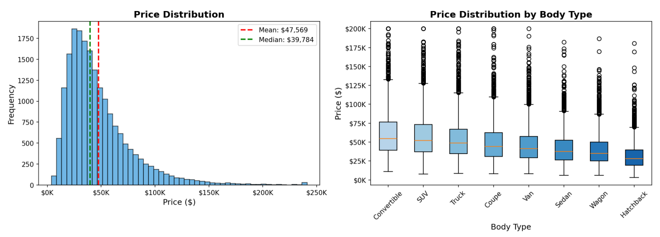

Fig. 5. Left: Overall price distribution showing right skew (mean $47,569, median $34,988). Right: Price variation by body type, with SUVs showing the highest median prices.

V. Discussion

A. Linear Regression Baseline Analysis

The Linear Regression model achieved an R^2 of 0.7034 and RMSE of $29,605, providing a meaningful baseline for comparison with ensemble methods. While significantly underperforming the tree-based models, this result demonstrates that linear relationships capture approximately 70% of the price variance in the dataset.

Several factors contribute to Linear Regression's lower performance relative to ensemble methods:

Non-linear Relationships: Vehicle pricing inherently involves non-linear interactions. For example, the depreciation rate is not constant—vehicles typically depreciate faster in early years, then stabilize. Linear Regression cannot capture these patterns without explicit polynomial feature engineering.

Feature Interactions: The price impact of features like body type, make, and engine size interact in complex ways. A V8 engine may command a premium in a truck but less so in a sedan. Tree-based methods automatically discover such interactions, while Linear Regression treats features independently.

High-Dimensional Categorical Encoding: The one-hot encoding of categorical features (particularly Make with 40+ manufacturers) creates a sparse, high-dimensional feature space. We addressed potential multicollinearity by using `drop='first'` in the OneHotEncoder, which eliminates the dummy variable trap and ensures numerical stability.

Scale Sensitivity: While StandardScaler normalizes numerical features, the binary one-hot encoded columns remain on a different scale. This scale imbalance can bias coefficient estimates, though the regularization effect of dropping one category per feature helps mitigate this issue.

1) Potential Improvements for Linear Models: To improve linear model performance beyond the baseline, the following modifications could be explored:

1) **Ridge Regression (L2 Regularization):** Add coefficient penalties to improve generalization:

$$\min_{\beta} \|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \tag{5}$$

2) **Lasso Regression (L1 Regularization):** Use L1 penalties for automatic feature selection by driving irrelevant coefficients to zero.
3) **Polynomial Features:** Explicitly model non-linear relationships through polynomial expansion of key numerical features.
4) **Log Transform Target:** Given the right-skewed price distribution, predicting $\log(\text{Price})$ may improve linear model performance by normalizing the target distribution.
5) **Feature Selection:** Remove highly correlated features (e.g., one of City/Highway fuel efficiency) to reduce redundancy and improve coefficient interpretability.

B. Ensemble Model Success

The strong performance of Gradient Boosting and Random Forest can be attributed to their inherent properties:

Non-linearity: Tree-based methods naturally capture non-linear relationships between features and price without explicit polynomial feature engineering.

Robustness to Multicollinearity: Decision trees split on individual features, making them less sensitive to correlated predictors than linear methods.

Automatic Feature Selection: Trees inherently perform feature selection by choosing the most informative splits, effectively ignoring uninformative features.

Handling of Mixed Types: Tree methods handle both numerical and categorical features naturally without requiring careful scaling.

C. Architecture Trade-offs

The hybrid PySpark/scikit-learn architecture successfully addressed Databricks Community Edition limitations while maintaining code organization. However, this approach has scalability implications:

Memory Constraints: Converting Spark DataFrames to Pandas requires the entire dataset to fit in driver memory. Our 19,461-record dataset (approximately 15MB) was well within limits, but larger datasets would require sampling or distributed alternatives.

Production Considerations: For production deployment on paid Databricks tiers, the architecture should be refactored to use native PySpark ML pipelines with MLflow model registry integration.

D. Business Implications

The Gradient Boosting model's MAE of $6,553 represents approximately 18.7% of the median vehicle price ($34,988). This level of accuracy enables several practical applications:

- **Seller Pricing Guidance:** Providing market-aligned listing price recommendations
- **Buyer Due Diligence:** Identifying potentially over/underpriced vehicles
- **Dealer Inventory Valuation:** Supporting bulk vehicle portfolio assessment
- **Insurance Valuations:** Assisting in replacement value estimates

VI. Conclusion

This paper presented a comprehensive machine learning pipeline for second-hand vehicle price prediction, implementing a medallion architecture on Databricks. Key achievements include:

1) Successful implementation of a Bronze $\rightarrow$ Silver $\rightarrow$ Gold data pipeline processing 24,198 records with rigorous quality controls, yielding 19,461 high-quality records for modeling
2) Gradient Boosting model achieving R^2 = 0.9090 and RMSE = $16,399, demonstrating strong predictive capability
3) Random Forest providing robust intermediate performance with R^2 = 0.8504, confirming ensemble method effectiveness
4) Linear Regression establishing a meaningful baseline with R^2 = 0.7034, capturing approximately 70% of price variance through linear relationships
5) A practical hybrid architecture addressing platform constraints while maintaining production patterns

The results demonstrate that ensemble tree-based methods significantly outperform linear approaches for vehicle price prediction, as expected given the inherently non-linear nature of automotive pricing. The Linear Regression baseline provides valuable context for quantifying the improvement gained from more sophisticated algorithms.

A. Future Work

Several directions merit further investigation:

- **Advanced Models:** Evaluate XGBoost, LightGBM, and neural network architectures for potential performance gains
- **Feature Engineering:** Incorporate market indices, seasonal adjustments, and geographic pricing variations

- **Model Interpretability:** Apply SHAP values to explain individual predictions and identify key pricing factors
- **Real-time Deployment:** Implement streaming inference pipeline for live pricing updates
- **Regularized Linear Models:** Implement Ridge/Lasso regression to potentially improve linear model performance beyond the OLS baseline

ACKNOWLEDGMENTS

The authors thank the ENSF 612 course instructors for guidance on big data engineering principles and the Databricks Community Edition platform for providing free cloud computing resources.

REFERENCES

[1] F. Hossein, "Used Vehicles for Sale Dataset," Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/farhanhossein/used-vehicles-for-sale

[2] M. Armbrust *et al.*, "Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3411–3424, 2020.

[3] Databricks, "Medallion Architecture," Databricks Documentation, 2024. [Online]. Available: https://docs.databricks.com/en/lakehouse/medallion.html

[4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[6] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] Apache Software Foundation, "PySpark Documentation," 2024. [Online]. Available: https://spark.apache.org/docs/latest/api/python/

[8] P. Lesani, S. M. Vieira, and J. M. C. Sousa, "Used Car Price Prediction using Machine Learning Techniques," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2020, pp. 1–6.

[9] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[10] D. A. Belsley, E. Kuh, and R. E. Welsch, *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. New York: Wiley, 1980.