Kelly Arseneau

Data Science Portfolio

M.S. in Applied Data Science

Syracuse University | June 2025

This portfolio showcases the depth, range, and real-world readiness of my data science skillset. Organized around the core competencies of Syracuse University's Applied Data Science program, each section reflects my ability to collect and structure data, extract insight, model outcomes, and communicate findings with clarity and purpose.

What sets this work apart is its grounding in authentic, complex problems — from modeling housing markets and F1 race outcomes to building end-to-end infrastructure for business intelligence. I've used R, Python, SQL, Spark, Power BI, and more to build solutions that are not only technically sound, but actionable.

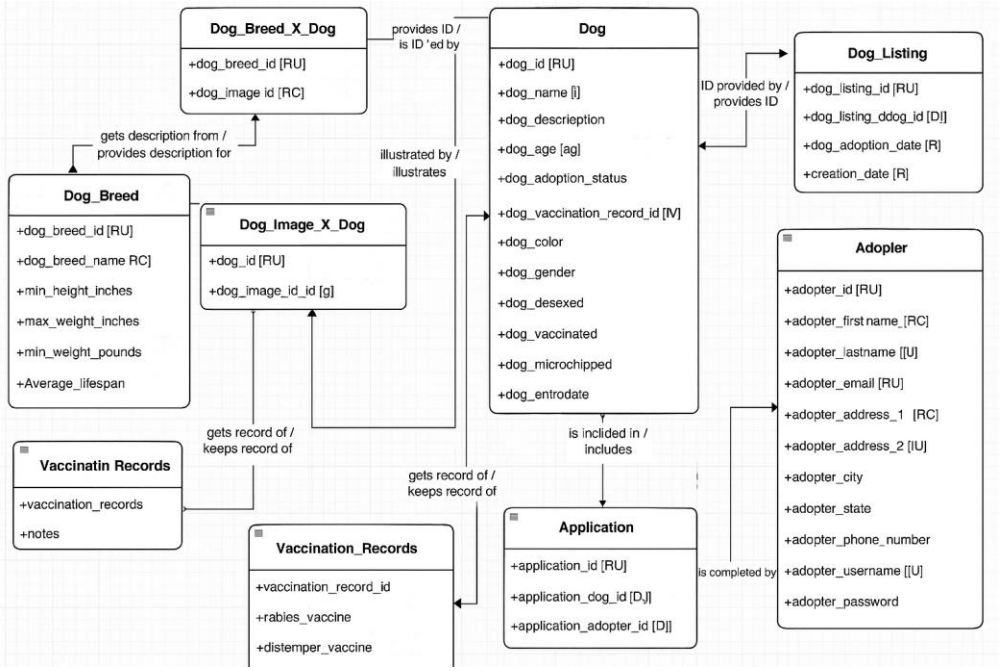
The result is a collection of projects that demonstrate not just what I know, but how I think creatively, critically, and ethically.

I. From chaos to cloud — all in a day's wrangling.

Data Engineering & Scalable Infrastructure: Designing efficient, scalable systems for data collection, storage, and access is foundational to real-world data science. These projects showcase my ability to build end-to-end data pipelines, model relational databases for business intelligence, and process millions of records using distributed computing tools like PySpark. Whether working with cloud-ready data or designing ETL workflows, I bring a deep understanding of how to structure data for performance, flexibility, and insight.

Because every pup deserves a forever home and a well- normalized schema.

When SQL Met Shelter Dogs: .Built a relational database schema from the ground up to support core operations of an adoption system. Designed tables and relationships using ER diagrams in Draw.io, then implemented the schema in PostgreSQL with SQL DDL. Included foreign keys to maintain data integrity and designed user roles to control access. Tested the system with realistic queries involving dogs, customers, and adoption records.



Schema Design: ER diagram showing relational structure of the dog adoption system, including foreign keys and normalized tables.

This project laid the groundwork for my understanding of performance-aware data modeling — a skill now embedded in my approach to larger data architecture and pipeline design.

Because 2 million rows don't clean themselves.

End-to-End ML Workflow Using PySpark: Tell me it can't be done, and I'll find a way to make it work. This project challenged me to process over 2 million records in PySpark, cleaning, joining, and transforming data on housing, pricing, and location at cloud scale.

I reworked my feature engineering approach midstream to maintain data integrity and avoid misleading correlations — reinforcing the importance of designing pipelines that are not just powerful, but reliable. What started as a domain-specific project became a lesson in building scalable, production-ready workflows that

```
from pyspark.sql.functions import substring

# Extract the first 3 digits of zip_code to create zip_region
df_cleaned = df_cleaned.withColumn("zip_region", substring(col("zip_code").cast("string"), 1, 3))

# Verify zip_region was created successfully
df_cleaned.select("zip_region").distinct().show(5)

# Show first 10 rows with zip_code and zip_region to verify correctness
df_cleaned.select("zip_region", "city", "price", "bed", "bath", "house_size").show(10, truncate=False)
```

```
+----------+
|zip_region|
+----------+
|       125|
|       124|
|       030|
|       088|
|       101|
+----------+
only showing top 5 rows

+----------+------+--------+---+----+----------+
|zip_region|city  |price   |bed|bath|house_size|
+----------+------+--------+---+----+----------+
|010       |Agawam|180000.0|2  |1   |676       |
|010       |Agawam|239900.0|3  |1   |1196      |
|010       |Agawam|525000.0|3  |3   |2314      |
|010       |Agawam|289900.0|3  |2   |1276      |
|010       |Agawam|275000.0|4  |2   |1732      |
|010       |Agawam|335000.0|6  |2   |1800      |
|010       |Agawam|384900.0|3  |2   |1476      |
|010       |Agawam|779900.0|4  |3   |2910      |
|010       |Agawam|199999.0|3  |2   |1968      |
|010       |Pelham|419000.0|4  |2   |1607      |
+----------+------+--------+---+----+----------+
only showing top 10 rows
```

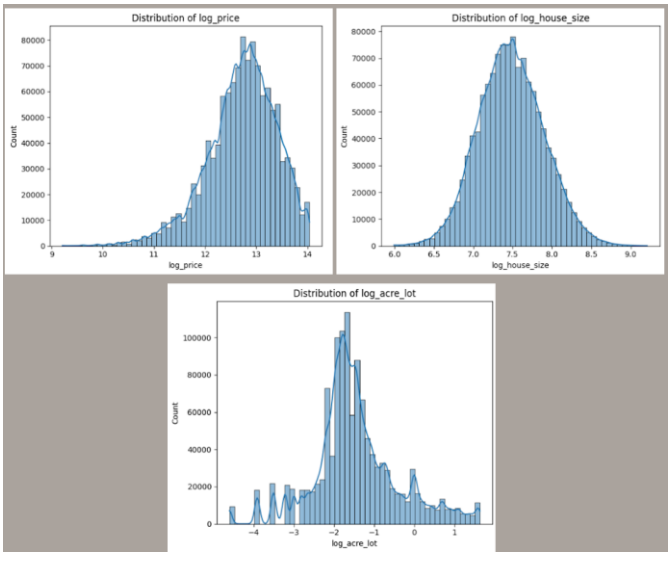
Feature Engineering: Created a `zip_region` variable by extracting digits from ZIP codes to support regional analysis.

```
# Initialize Spark Session
spark = SparkSession.builder.appName("RealEstatePricePrediction").config
("spark.sql.execution.arrow.enabled", "true").getOrCreate()

# Load dataset from the current directory
file_path = "realtor-data.csv"

# Check if the file exists before loading (optional but safe)
import os
if not os.path.exists(file_path):
    raise FileNotFoundError(f"❌ '{file_path}' not found in current directory.")

# Load the dataset using Spark
df = spark.read.csv(file_path, header=True, inferSchema=True)
df.printSchema()
df.show(5)
```

```
root
 |-- brokered_by: integer (nullable = true)
 |-- status: string (nullable = true)
 |-- price: double (nullable = true)
 |-- bed: integer (nullable = true)
 |-- bath: integer (nullable = true)
 |-- acre_lot: double (nullable = true)
 |-- street: integer (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- zip_code: integer (nullable = true)
 |-- house_size: integer (nullable = true)
 |-- prev_sold_date: string (nullable = true)
```

Cloud Workflow: Built and executed PySpark transformations in Google Colab for scalable processing.

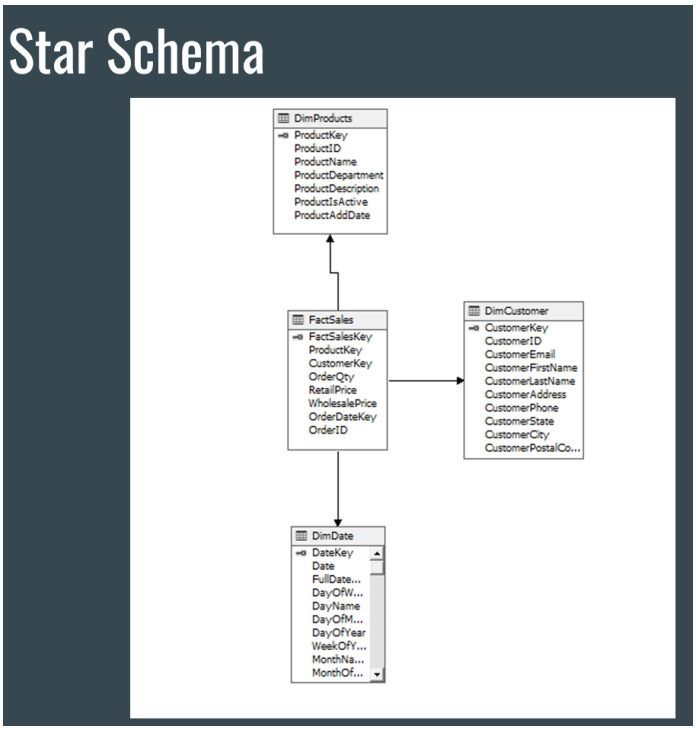
prioritize both accuracy and interpretability.



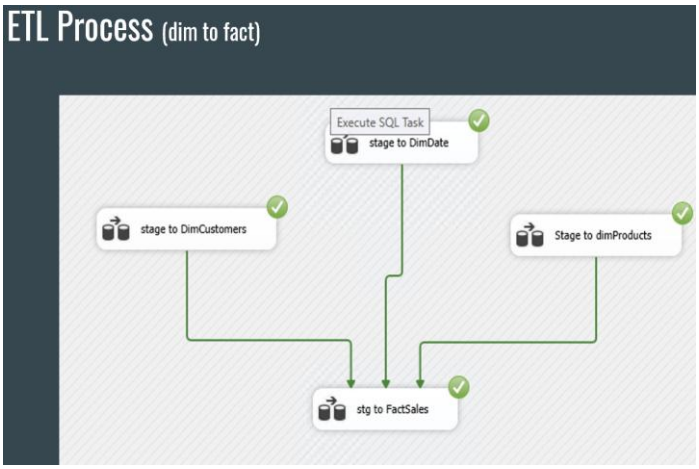
Feature Transformation: *Applied log transformations to normalize distributions and stabilize model variance*

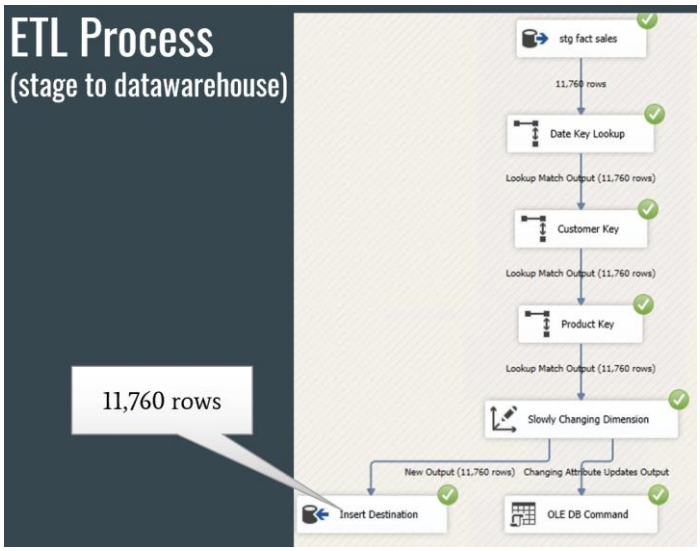
Cleaning up raw data and teaching it to behave

Data Warehouse Design & ETL Development: Architected a full-scale data warehouse solution from raw operational datasets, including retail and streaming service data. Designed dimensional models with star schemas, implemented staging layers, and developed a robust end-to-end ETL pipeline.
Built with performance and flexibility in mind, this project sharpened my ability to create enterprise-ready data infrastructure—pipelines and models that power BI tools and support fast, accurate decision-making.



Dimensional Modeling: Designed a star schema to separate fact and dimension tables, optimizing structure for BI performance and flexibility..





ETL Workflow – Dim to Fact: *Loaded dimension tables before facts to ensure referential integrity during staging.*

ETL Workflow – Stage to Warehouse: *Inserted fully staged data into the warehouse using surrogate keys and SCD techniques.*

High Level Model

Business Process Name	Fact Table	Fact Grain Type	Granulairty	Facts
sales reporting for fudgemart	sales_fact	transaction	one row per order detail	quantity, retail price
sales reporting for fudgeflix	sales_fact	transaction	one row per order detail	billed amount, count of customers
profit reporting for fudgemart	profit analysis fact	transaction	one row for product and profit margin	quantity, retail price, wholesale price
order fufillment for fudgemart	order fufillment fact	transaction	one row per order detail	order lag: order date, shipped date
customer usage for fudgeflix	customer usage fact	transaction	one row per order detail	order usage time: returned date, shipped date
pay roll process analysis fudgemart	pay roll process fact	assignment	one row for employee pay roll	timesheet hourly rate, timesheet hours
customer product reviews fudgemart	customer review fact	assignment	one row for customer review	review date, review stars,
customer product reviews fudgeflix	customer review fact	assignment	one row for customer review	rating

Business Process Mapping: *Modeled data marts across retail and streaming services, defining fact grains and metrics to support flexible analysis. .*

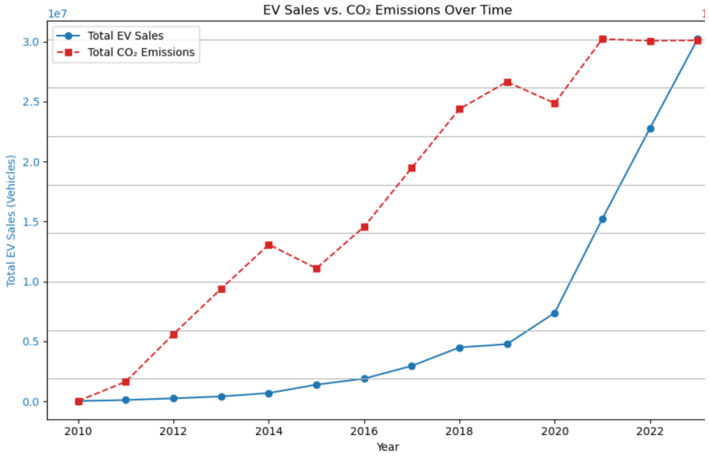
II. Connecting the dots so others don't have to squint

Insight-Driven Problem Solving with the Full Data Science Life Cycle: Creating impact with data means more than wrangling numbers — it's about uncovering patterns, spotting opportunities, and translating messy inputs into meaningful, real-world insight. The projects in this section reflect my ability to ask the right questions, build context-aware models, and generate results that matter — whether that means reducing emissions, predicting race outcomes, or identifying market inefficiencies. From exploratory analysis to model tuning, these examples show how I turn information into action.

Kicking gas one insight at a time

Impact Modeling Using Multi-Source Data Integration: Analyzed CO_2 emissions alongside EV adoption data to generate insight into environmental trends and policy impact. Combined real-time emissions APIs with historical adoption data to reveal regional patterns, exploring the effectiveness of green tech on actual outcomes. Rather than just visualizing trends, I contextualized them, turning raw data into actionable environmental intelligence.

```python
import matplotlib.pyplot as plt

#  1: Summarize EV Sales and Annual CO₂ Emissions Over Time
ev_sales_over_time = ev_sales_historical.groupby('Year')['value'].sum().reset_index()
co2_over_time = ev_sales_historical.groupby('Year')['Annual CO₂ emissions'].sum().reset_index()

# Merge summaries into a single table
summary_over_time = pd.merge(ev_sales_over_time, co2_over_time, on='Year')
summary_over_time.rename(columns=
    {'value': 'Total EV Sales', 'Annual CO₂ emissions': 'Total CO₂ Emissions'}, inplace=True)

# Display the summary table
print("Summary of EV Sales and CO₂ Emissions Over Time:")
print(summary_over_time)

#  2: Visualize EV Sales and CO₂ Emissions Over Time
fig, ax1 = plt.subplots(figsize=(10, 6))
```



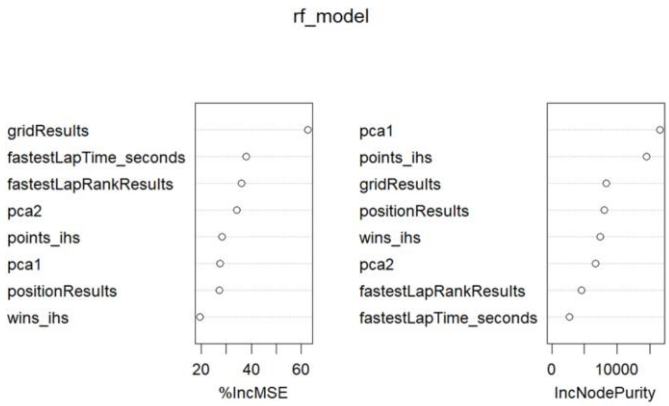
Aggregated and merged EV and CO_2 data over time to support time-series analysis—part of the modeling prep phase in the data science life cycle.

***Time Series Insight**:Compared long-term EV adoption and CO_2 emissions trends to surface counterintuitive insight: despite skyrocketing EV sales, emissions continue rising—revealing a gap between tech adoption and environmental impact.T*

Predicting Podiums Like a Data-Fueled Pit Crew

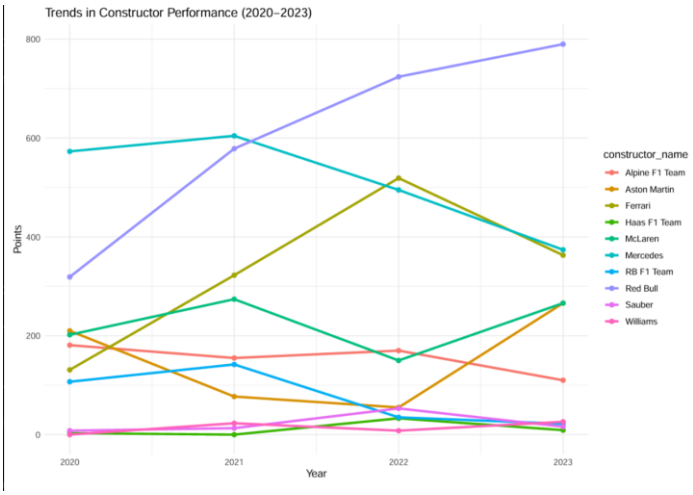
Performance Prediction from High-Variance Sports Data: Modeled and interpreted team success in Formula 1 using a multi-faceted dataset of race results, driver stats, and constructor variables. Extracted insights by isolating predictive indicators (e.g., driver consistency, team spending, track types) and translating them into strategic context. This project illustrates how actionable insight can emerge from highly variable data environments—whether in sports, finance, or beyond.

Random Forest Variable Importance Plot



Top predictive features identified by the Random Forest model, offering insight into which variables most influenced team success.

Constructor Performance Line Plot (‘20–’23)



Historical performance trends of F1 constructors (2020–2023), used to contextualize predictive patterns and identify high-variance teams.

```r
# Load races data
races <- read_csv("C:/Users/kelly/Documents/Formula 1/races.csv")

# Check the structure of the races dataframe
print("Structure of races dataframe:")
print(str(races))

# Join races with merged_data to include the date column
merged_data <- merged_data %>%
  left_join(races %>% select(raceId, date), by = "raceId")

# Check if 'date' column exists and print first few values
print("First few entries in the date column after join:")
print(head(merged_data$date))

# Identify problematic date values (using correct parser for m/d/y)
problematic_dates <- merged_data$date[is.na(mdy(merged_data$date))]
print("Problematic date values:")
print(problematic_dates)

# Convert 'date' column to Date type
merged_data <- merged_data %>%
  mutate(date = mdy(date))
```

Prepping F1 race data in R: Loaded and joined multiple CSVs, identified malformed date values, and converted them to proper datetime format for time-series analysis.

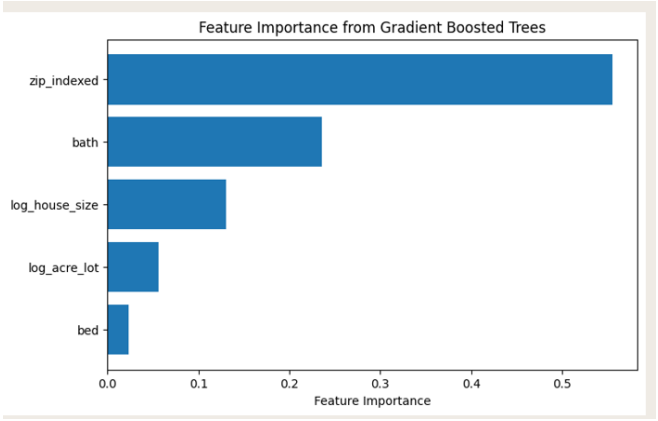
```r
# Prepare the data for time series analysis
time_series_data <- merged_data %>%
  mutate(year = year(date)) %>%
  group_by(year, constructorId) %>%
  summarise(total_points = sum(points, na.rm = TRUE), .groups = "drop") %>%
  left_join(constructors, by = "constructorId")

# Check structure
print("Structure of time_series_data dataframe:")
print(str(time_series_data))

# Plot the time series data
ggplot(time_series_data, aes(x = year, y = total_points, color = name)) +
  geom_line() +
  labs(title = "Constructors' Performance Over Time",
       x = "Year",
       y = "Total Points",
       color = "Constructor") +
  theme_minimal()
```

Grouped constructor results by year to compute total points, then visualized performance over time using `ggplot2`.

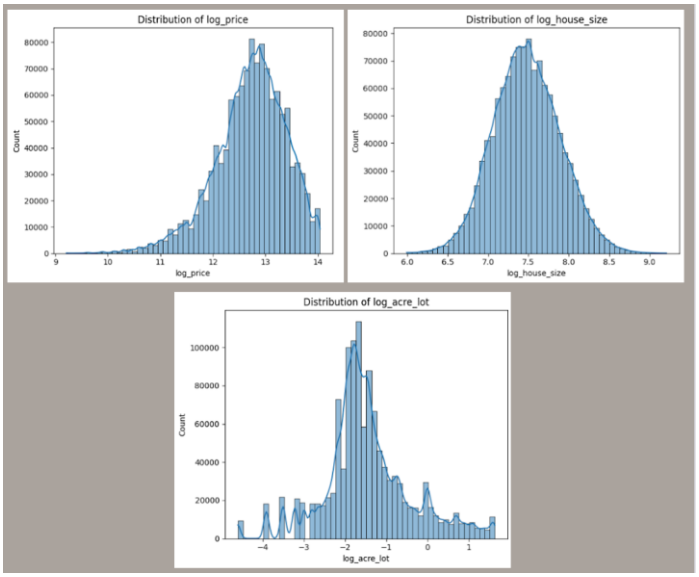
Feature engineering that survived 2 million records and lived to tell the tale

Predictive Feature Engineering & Scalable Pipeline Development:

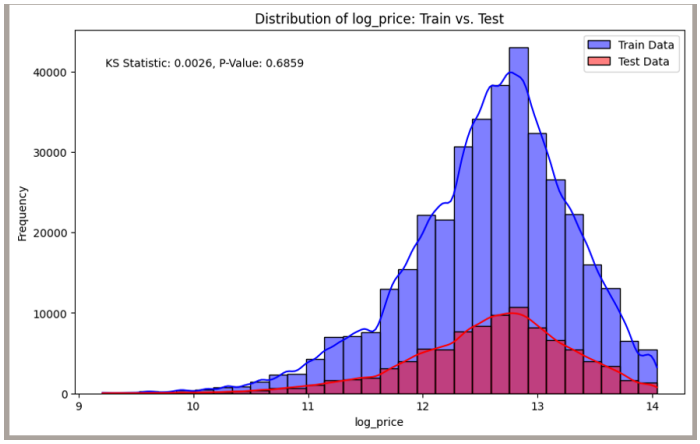
Created an interpretable, end-to-end predictive pipeline to extract meaning from over 2 million housing records. This project focused on distilling key signals (e.g., location + structure features) to predict outcomes with real-world usability. I fine-tuned feature selection and engineered custom variables like `zip_region`, producing insights not just about prices — but about why certain homes performed differently in the market. I also designed the train/test split to prevent data leakage and overfitting, keeping the evaluation honest and the predictions reliable.



GBT Feature Impact: *Showed model interpretability by ranking features based on contribution to predictive performance.*



Log-Transformed Feature Distributions: *Log scaling normalized skewed features like price, house size, and lot size for better model fit.*



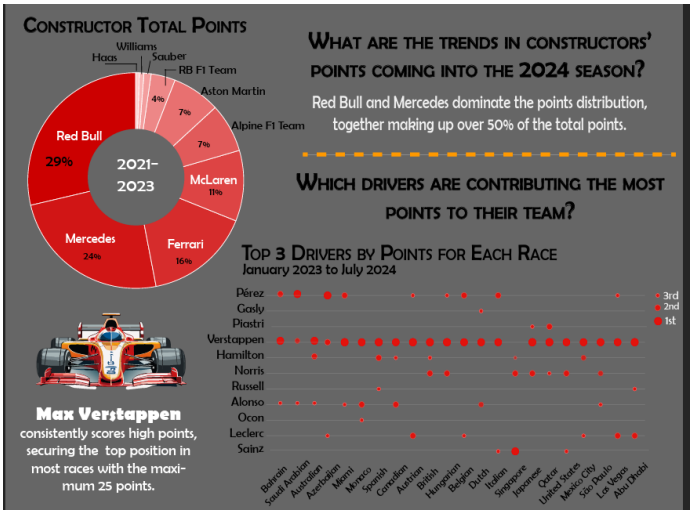
Train vs. Test Distribution: *The log price distribution is consistent across training and test sets, supporting a valid model split.*

III. The Future Called — It Liked My Visualization

Turning Complex Models into Strategic Storytelling: In a world flooded with metrics, the ability to visualize and predict isn't just technical—it's transformative. These projects show how I blend forecasting with visual design to make insights accessible, memorable, and actionable. From modeling high-variance outcomes to crafting visuals that resonate with real audiences, I focus on clarity, context, and impact. My work balances analytical rigor with visual storytelling. These projects highlight how I turn models and metrics into strategic narratives; because seeing is understanding.

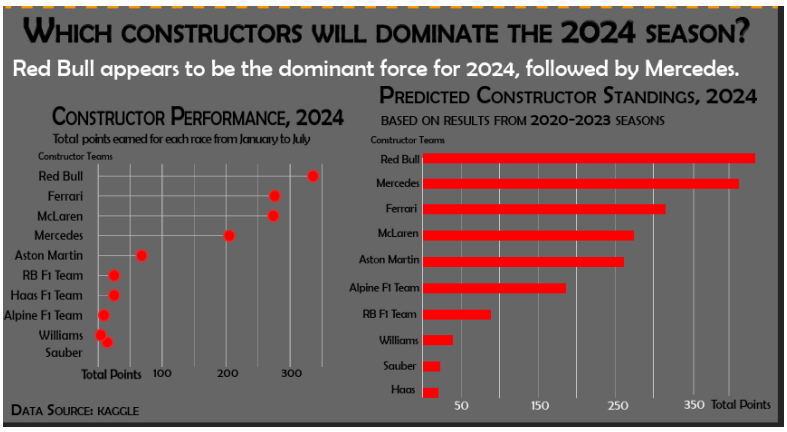
Pit Stops, Plotlines, and Predictions

Visualization-Driven Forecasting of 2024 F1 Constructors: Using historical performance data from 2019–2023 and midseason results from 2024, I built a predictive framework to forecast which F1 constructors would dominate the current season. After subsetting and cleaning over 600 observations, I used R and Plotly to visualize team performance trends and driver contributions, then applied ensemble modeling to support the projected standings. The final product was a polished infographic poster designed for F1 fans and fantasy players, combining statistical rigor with visual storytelling to surface season-defining patterns.




```r
library(ggplot2)
library(dplyr)

# Summarize constructor points across 2021-2023 seasons
constructor_summary <- constructorPerformance %>%
  filter(year >= 2021 & year <= 2023) %>%
  group_by(names) %>%
  summarise(total_points = sum(totalPoints)) %>%
  mutate(percent = round(100 * total_points / sum(total_points), 1),
         label = paste0(names, "\n", percent, "%"))

# Create pie chart to visualize constructor dominance
ggplot(constructor_summary, aes(x = "", y = total_points, fill = names)) +
  geom_col(width = 1, color = "white") +
  coord_polar(theta = "y") +
  theme_void() +
  labs(title = "Constructor Total Points\n2021-2023") +
  geom_text(aes(label = label),
            position = position_stack(vjust = 0.5), size = 4)
```

Team & Driver Insights: *Visualized constructor and driver point contributions from 2021–2023 to uncover performance trends entering the 2024 F1 season.*




```r
```{r}
library(ggplot2)
library(dplyr)

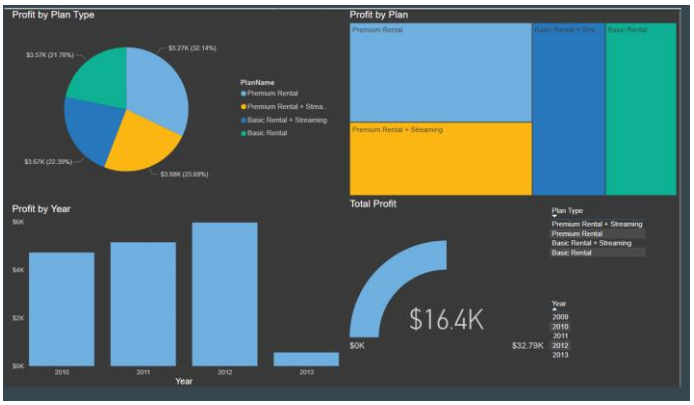
# Total constructor points from Jan to July 2024
performance_2024 <- df_2024 %>%
  group_by(constructorRef) %>%
  summarise(total_points = sum(points)) %>%
  arrange(desc(total_points))

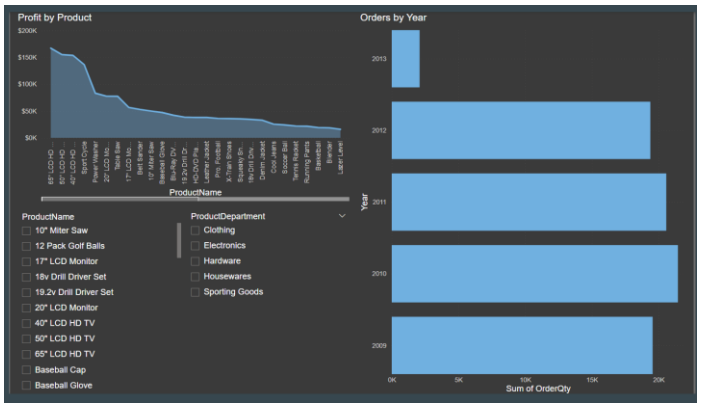
# Horizontal bar chart for actual 2024 performance
ggplot(performance_2024, aes(x = reorder(constructorRef, total_points), y = total_points)) +
  geom_col(fill = "red") +
  coord_flip() +
  labs(title = "Constructor Performance, 2024",
       x = "Constructor Teams", y = "Total Points") +
  theme_minimal()
```

Performance Forecasting: *Combined historical data and predictive modeling to visualize projected constructor standings for the 2024 season.*

Dashboard Domination: Where Pie Charts and Profits Collide

Data Visualization & Exploratory Analysis with Power BI This project highlights my ability to build insightful dashboards using Power BI to uncover business trends across streaming and retail divisions. I visualized profit by year, product, and plan type using pie charts, bar graphs, tree maps, and line plots—demonstrating how to craft clear, dynamic views for decision-makers.




```

Multiview Dashboard: *Combined pie, bar, treemap, and gauge visuals to uncover profitability trends across streaming plans and years. Highlighted Premium Rental as the top-earning plan and 2012 as the peak year, showcasing the power of layered BI storytelling.*

Sales Insights Dashboard: *Visualized product-level profitability and order frequency using line and bar charts, enabling actionable insights by department and time. Highlighted the dominance of 65" LCD TVs and the sales boom in 2010.*

Combined Profit Dashboard: *Merged multi-source data into a single view showing total profit trends, year-over-year breakdowns, and quarterly performance. Designed with cross-filtering and slicers to enable executive decision-making at a glance.*

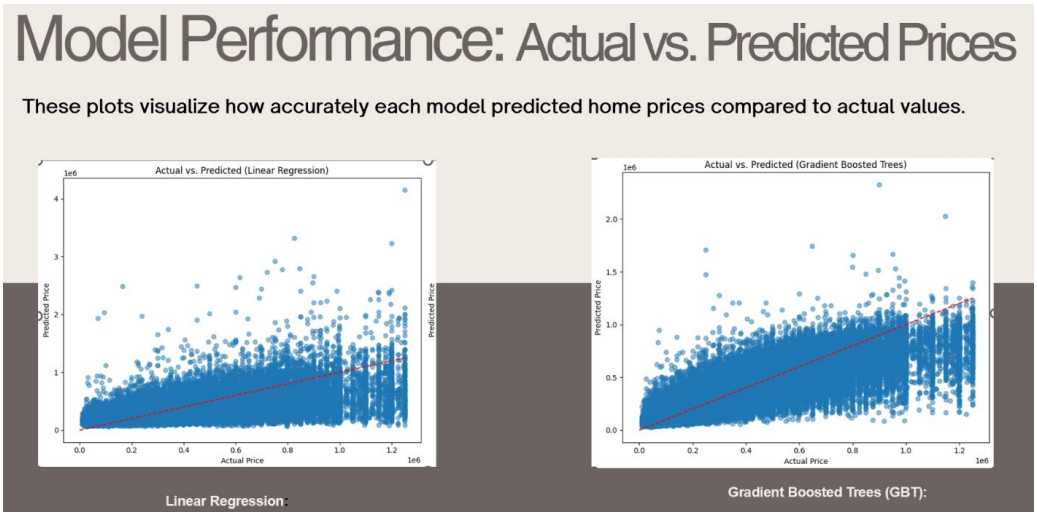
More than just reporting, these dashboards were used to explore underlying business drivers, compare performance across departments, and identify time-based trends. This project sharpened my skills in exploratory data analysis (EDA), visual communication, and interactive dashboard design—core tools for analysts and stakeholders making data-informed decisions.

Turning Property Specs into Price Tags

Predictive Modeling with Linear Regression and Gradient Boosted Trees: This project involved building interpretable and accurate regression models to predict home prices based on features like square footage, number of bathrooms, lot size, and zip region. I explored feature distributions, applied log transformations to normalize skewed data, and carefully validated model performance on train/test splits to avoid overfitting.

Visualizations played a key role throughout the process — from histograms and feature importance charts to predicted-vs-actual scatterplots that revealed performance patterns and model fit. I compared linear regression and gradient boosted trees, balancing interpretability with accuracy, and used model diagnostics (like KS statistics and distribution comparisons) to assess generalizability.

This project sharpened my ability to visualize relationships, validate model performance, and communicate insight in a way that bridges the gap between statistical rigor and stakeholder clarity.



Model Accuracy Comparison: *Compared predicted vs. actual home prices across both linear and boosted tree models.*

```python
import matplotlib.pyplot as plt
from pyspark.sql.functions import exp

# Visualize actual vs. predicted prices with error bars
prediction_pd = predictions_dollars.select("actual_price", "predicted_price",
                                           "error_dollars").toPandas()

plt.figure(figsize=(10, 6))
plt.scatter(prediction_pd["actual_price"], prediction_pd["predicted_price"],
            alpha=0.5)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs. Predicted House Prices (Linear Regression)")

# Add error bars
for i in range(len(prediction_pd)):
    plt.plot([prediction_pd["actual_price"][i], prediction_pd["actual_price"][i]],
             [prediction_pd["predicted_price"][i] - prediction_pd["error_dollars"]
              [i], prediction_pd["predicted_price"][i] +
              prediction_pd["error_dollars"][i]],
              color="gray", linewidth=0.5)

plt.plot([0, prediction_pd["actual_price"].max()], [0, prediction_pd
 ["actual_price"].max()], color="red", linestyle="--")

plt.show()
```

Visualized actual vs. predicted home prices using PySpark output and matplotlib, with error bars to show prediction uncertainty. This helped assess model accuracy and identify outliers.

IV. Code Is My Co-Pilot

In today's data landscape, scripting isn't just a support skill — it's the engine behind exploration, modeling, and automation. These projects show how I used R and Python to clean data, explore relationships, and generate meaningful output. Whether I was building apps or running models, the focus was always on clarity — not just code.

If you've made it this far, thanks for sticking with it. Behind every line of code and chart in this portfolio is a commitment to clarity, impact, and integrity. Whether I was optimizing data pipelines or forecasting complex systems, the goal was never just to make something work — it was to make it matter.

Every project in here reflects real decisions I had to make — what to clean, what to model, how to tell the story, and when to step back and keep the work honest. From pipelines to predictions to visual storytelling, my approach stays the same: keep it clean, make it useful, and build something that holds up.

Contact

kelly.noel.ds@gmail.com

github.com/kelly12201984

linkedin.com/in/kelly-arseneau-9459b1273