02_analysis

February 5, 2026

[5]:
```
%pip install -r ../requirements.txt
```

```
Note: you may need to restart the kernel to use updated packages.
```

[6]:
```
from pathlib import Path
import sys

ROOT = Path('..').resolve()
SRC = ROOT / 'src'
sys.path.append(str(SRC))

import numpy as np
import pandas as pd
import plotly.express as px
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

import engine_atlas.data_processing as dp

DATA_PATH = ROOT / 'data' / 'Car Dataset 1945-2020.csv'
print('DATA_PATH:', DATA_PATH)

graphs_dir = ROOT / 'assets' / 'graphs'
graphs_dir.mkdir(parents=True, exist_ok=True)

def save_fig(fig, name):
    out_path = graphs_dir / f"{name}.png"
    fig.write_image(out_path)
    return fig

parquet_path = ROOT / 'data' / 'processed' / 'engine_atlas_cleaned.parquet'
required_cols = {
    'year',
    'engine_hp',
    'acceleration_0_100_km_h_s',
    'number_of_cylinders',
    'mixed_fuel_consumption_per_100_km_l',
```

```
    'co2_emissions_g_km',
    'engine_signature',
    'hp_per_liter',
}
if parquet_path.exists():
    df = pd.read_parquet(parquet_path)
    if 'acceleration_0_100_km_h_' in df.columns and 'acceleration_0_100_km_h_s'␣
 ↪not in df.columns:
        df = df.rename(columns={'acceleration_0_100_km_h_':␣
 ↪'acceleration_0_100_km_h_s'})
    if not required_cols.issubset(df.columns):
        df = dp.clean_engine_data(str(DATA_PATH))
        df.to_parquet(parquet_path, index=False)
else:
    df = dp.clean_engine_data(str(DATA_PATH))
    df.to_parquet(parquet_path, index=False)

df.head()
```

[6]:

	id_trim	make	model	generation	year_from	year_to	series	trim	\
0	1	AC	ACE	1 generation	1993.0	2000.0	Cabriolet	3.5 MT	
1	2	AC	ACE	1 generation	1993.0	2000.0	Cabriolet	4.6 MT	
2	3	AC	ACE	1 generation	1993.0	2000.0	Cabriolet	4.9 AT	
3	4	AC	ACE	1 generation	1993.0	2000.0	Roadster	2.9 AT	
4	5	AC	ACE	1 generation	1993.0	2000.0	Roadster	2.9 MT	

	body_type	load_height_mm	…	battery_capacity_kw_per_h	electric_range_km	\
0	Cabriolet	None	…	NaN	NaN	
1	Cabriolet	None	…	NaN	NaN	
2	Cabriolet	None	…	NaN	NaN	
3	None	None	…	NaN	NaN	
4	None	None	…	NaN	NaN	

	charging_time_h	year	bore_mm	stroke_mm	displacement_l	hp_per_liter	\
0	NaN	1993.0	83.0	NaN	3.505	100.998573	
1	NaN	1993.0	90.0	NaN	4.601	70.854162	
2	NaN	1993.0	101.0	NaN	4.942	52.610279	
3	NaN	1993.0	NaN	NaN	NaN	NaN	
4	NaN	1993.0	NaN	NaN	NaN	NaN	

	engine_signature	balanced_score
0	AC Gasoline V-type 8.0 3.5L	1.792864
1	AC Gasoline V-type 8.0 4.6L	1.642758
2	AC Gasoline V-type 8.0 4.94L	0.588031
3	AC L	NaN
4	AC L	NaN

```
[5 rows x 85 columns]
```

1 Engine Atlas - Analysis

This notebook answers 10+ analytical questions with Plotly visualizations.

1.1 Results & Insights

1.1.1 1. Average Horsepower Over Time

Average horsepower shows a long-term upward trajectory across the dataset.

- 1940s–1950s: roughly 80–120 HP
- 1960s: peak near 200–250 HP during the muscle car era
- 1970s: decline to ~100–130 HP, consistent with emissions regulations and the oil crisis
- 1980 onward: steady growth to 200+ HP in modern vehicles

Conclusion: modern vehicles meet or exceed 1960s peak power while delivering higher efficiency.

1.1.2 2. Acceleration (0–100 km/h) Over Time

Acceleration performance improves markedly over time.

- Early vehicles commonly required 20–40 seconds
- By the 1970s, averages stabilized around ~14 seconds
- Contemporary models achieve ~8–10 seconds or better

Conclusion: acceleration has improved dramatically despite rising vehicle mass, driven by turbocharging, AWD, and electrification.

1.1.3 3. Cylinder Count Trend

The median cylinder count shifts toward smaller engines.

- 1950s–1960s: 6–8 cylinders dominate
- Post-1980: median shifts strongly toward 4 cylinders
- Spikes reflect low-volume performance models

Conclusion: the data indicates sustained downsizing, supported by forced induction and hybrid assistance.

1.1.4 4. Horsepower vs Fuel Consumption

Horsepower and fuel consumption are positively correlated.

- Higher HP generally implies higher fuel use
- Large spread at mid-HP levels indicates substantial technology effects

Conclusion: power alone no longer determines efficiency.

1.1.5 5. Horsepower vs CO2 Emissions

Emissions increase with power, but the relationship is not uniform.

- High-HP vehicles often emit 250–400 g/km
- Significant variation exists at similar HP levels

Conclusion: drivetrain and engine technologies are primary determinants of emissions outcomes.

1.1.6 6. Brands Leading in Median Horsepower

Median horsepower is led by hypercar and ultra-luxury manufacturers.

- Rimac (clear outlier)
- Koenigsegg, Vector, Bugatti, Lamborghini
- Saleen, Pagani, Maybach, McLaren
- Bentley and Ferrari

Conclusion: extreme-performance brands dominate median horsepower rankings.

1.1.7 7. Brands Leading in Efficiency (Lowest Fuel Consumption)

Efficiency leadership is concentrated among compact and economy-focused manufacturers.

- Bajaj, Perodua, DS, Changan, Hafei
- Smart, Dacia, Jiangnan, Mini, Daihatsu
- Followed by Skoda, SEAT, Fiat, Peugeot, Citroen, BYD, Vauxhall

Conclusion: efficiency leadership aligns with smaller vehicles and cost-sensitive segments.

1.1.8 8. Engine Type Comparison (HP vs Fuel)

Engine types exhibit clear trade-offs between power and fuel use.

- Electric: highest HP potential with near-zero fuel consumption
- Hybrid: strong balance of power and efficiency
- Diesel: moderate HP with better economy
- Gasoline: widest spread and highest average fuel use

Conclusion: the overall balance ranks as Electric $\rightarrow$ Hybrid $\rightarrow$ Diesel $\rightarrow$ Gasoline.

1.1.9 9. Top 10 Fastest Trims

The fastest trims are dominated by modern electric and high-performance powertrains.

- Rimac electric hypercars
- Tesla performance trims
- Bugatti W16
- Lamborghini V12
- Ferrari hybrid V8

Conclusion: electric torque increasingly defines acceleration benchmarks.

1.1.10 10. Best Power Density (HP per Liter)

Power density highlights the efficiency of modern small-displacement engines.

- Mazda rotary
- Mercedes AMG 2.0L
- Jaguar / Land Rover hybrid 2.0L
- Modern turbocharged inline-4 engines

Conclusion: small turbocharged and hybrid engines outperform large displacement engines on HP/L.

1.1.11 11. Engine Clustering (PCA + KMeans)

Unsupervised clustering separates vehicles by technological class.

- Economy: low HP, low fuel
- Conventional ICE: mid HP, moderate fuel
- Advanced: hybrid and modern turbo
- Performance / Electric: very high HP

Conclusion: the clusters naturally reflect distinct technology regimes.

Overall Summary. Horsepower has doubled over the long term, acceleration has improved drastically, and the market has shifted from large-displacement engines to turbocharged I4s and electrified powertrains. EVs break traditional power–efficiency trade-offs, hybrids occupy the middle ground, modern 2.0L engines rival historic V8s, and the market is polarized between efficiency-oriented and performance-oriented brands.

These results align with broader trends toward electrification and intelligent mobility, where software control, energy management, and advanced drivetrains reshape the performance–efficiency frontier.

1.2 Questions + Charts

1. How did average horsepower change over time?
2. How did acceleration (0–100) change over time?
3. Are engines shifting to fewer cylinders over time?
4. What is the horsepower vs fuel consumption tradeoff?
5. What is the horsepower vs CO2 tradeoff?
6. Which brands lead in median horsepower?
7. Which brands lead in efficiency?
8. How do engine types compare on HP and fuel consumption?
9. What are the top 10 fastest trims?
10. Which engines have the best power density (HP per liter)?
11. What engine families emerge from clustering?

[7]:
```
hp_trend = df.dropna(subset=['year', 'engine_hp']).groupby('year',␣
↪as_index=False)['engine_hp'].mean()
fig = px.line(hp_trend, x='year', y='engine_hp', title='Average Horsepower Over␣
↪Time')
```

```
save_fig(fig, 'avg_horsepower_over_time')
fig
```

[8]:
```
accel_trend = df.dropna(subset=['year', 'acceleration_0_100_km_h_s']).
 ↪groupby('year', as_index=False)['acceleration_0_100_km_h_s'].mean()
fig = px.line(accel_trend, x='year', y='acceleration_0_100_km_h_s',␣
 ↪title='Average 0-100 km/h Over Time')
save_fig(fig, 'avg_0_100_over_time')
fig
```

[9]:
```
cyl_trend = df.dropna(subset=['year', 'number_of_cylinders']).groupby('year',␣
 ↪as_index=False)['number_of_cylinders'].median()
fig = px.line(cyl_trend, x='year', y='number_of_cylinders', title='Median␣
 ↪Cylinders Over Time')
save_fig(fig, 'median_cylinders_over_time')
fig
```

[10]:
```
tradeoff = df.dropna(subset=['engine_hp',␣
 ↪'mixed_fuel_consumption_per_100_km_l'])
fig = px.scatter(tradeoff, x='engine_hp',␣
 ↪y='mixed_fuel_consumption_per_100_km_l', title='HP vs Fuel Consumption')
save_fig(fig, 'hp_vs_fuel_consumption')
fig
```

[11]:
```
co2_tradeoff = df.dropna(subset=['engine_hp', 'co2_emissions_g_km'])
fig = px.scatter(co2_tradeoff, x='engine_hp', y='co2_emissions_g_km', title='HP␣
 ↪vs CO2 Emissions')
save_fig(fig, 'hp_vs_co2_emissions')
fig
```

[12]:
```
hp_by_brand = df.dropna(subset=['make', 'engine_hp']).groupby('make',␣
 ↪as_index=False)['engine_hp'].median().sort_values('engine_hp',␣
 ↪ascending=False).head(20)
fig = px.bar(hp_by_brand, x='engine_hp', y='make', orientation='h', title='Top␣
 ↪20 Brands by Median HP')
save_fig(fig, 'top_20_brands_by_median_hp')
fig
```

[13]:
```
fuel_by_brand = df.dropna(subset=['make',␣
 ↪'mixed_fuel_consumption_per_100_km_l']).groupby('make',␣
 ↪as_index=False)['mixed_fuel_consumption_per_100_km_l'].median().
 ↪sort_values('mixed_fuel_consumption_per_100_km_l', ascending=True).head(20)
fig = px.bar(fuel_by_brand, x='mixed_fuel_consumption_per_100_km_l', y='make',␣
 ↪orientation='h', title='Top 20 Brands by Efficiency')
save_fig(fig, 'top_20_brands_by_efficiency')
fig
```

[14]:
```
engine_compare = df.dropna(subset=['engine_type', 'engine_hp'])
fig = px.box(engine_compare, x='engine_type', y='engine_hp', title='Engine Type␣
↪Comparison: HP')
save_fig(fig, 'engine_type_hp_comparison')
fig
```

[15]:
```
engine_compare_fuel = df.dropna(subset=['engine_type',␣
↪'mixed_fuel_consumption_per_100_km_l'])
fig = px.box(engine_compare_fuel, x='engine_type',␣
↪y='mixed_fuel_consumption_per_100_km_l', title='Engine Type Comparison: Fuel␣
↪Consumption')
save_fig(fig, 'engine_type_fuel_consumption_comparison')
fig
```

[16]:
```
fastest = df.dropna(subset=['acceleration_0_100_km_h_s']).nsmallest(10,␣
↪'acceleration_0_100_km_h_s')
fig = px.bar(fastest, x='acceleration_0_100_km_h_s', y='engine_signature',␣
↪orientation='h', title='Top 10 Fastest Trims')
save_fig(fig, 'top_10_fastest_trims')
fig
```

[17]:
```
density = df.dropna(subset=['hp_per_liter']).nlargest(10, 'hp_per_liter')
fig = px.bar(density, x='hp_per_liter', y='engine_signature', orientation='h',␣
↪title='Top 10 Power Density (HP/L)')
save_fig(fig, 'top_10_power_density')
fig
```

[18]:
```
features = ['engine_hp', 'acceleration_0_100_km_h_s',␣
↪'mixed_fuel_consumption_per_100_km_l', 'number_of_cylinders']
cluster_df = df[features].dropna()

if cluster_df.empty:
    print('No rows after dropna; reloading from raw CSV and rechecking.')
    df = dp.clean_engine_data(str(DATA_PATH))
    cluster_df = df[features].dropna()

if cluster_df.empty:
    print('Still empty after reload. Non-null counts:')
    print(df[features].notna().sum())
else:
    scaled = StandardScaler().fit_transform(cluster_df)
    labels = KMeans(n_clusters=4, random_state=42, n_init='auto').
↪fit_predict(scaled)
    coords = PCA(n_components=2, random_state=42).fit_transform(scaled)
    cluster_df = cluster_df.copy()
    cluster_df['cluster'] = labels
    cluster_df['pca_1'] = coords[:, 0]
```

```
    cluster_df['pca_2'] = coords[:, 1]
    fig = px.scatter(cluster_df, x='pca_1', y='pca_2', color='cluster',␣
↪title='Engine Clusters (PCA)')
    save_fig(fig, 'engine_clusters_pca')
    fig
```