1_eda

August 7, 2025

[ ]:
```
#!/usr/bin/env python
# coding: utf-8

# %% [markdown]
# ## Exploratory Data Analysis for Thermal Conductivity Prediction
#
# ## 1. Project Objective
#
# This notebook marks the first step in an end-to-end machine learning project␣
 ↪to predict the thermal conductivity of inorganic materials. The primary goal␣
 ↪of this initial phase-Exploratory Data Analysis (EDA)-is to thoroughly␣
 ↪understand the dataset's structure, identify potential data quality issues,␣
 ↪and uncover key relationships that will inform our subsequent modeling␣
 ↪strategies.
#
# ## 2. Business Context
#
# In materials science, discovering new materials with optimal thermal␣
 ↪properties is critical for developing next-generation electronics,␣
 ↪batteries, and energy systems. However, the traditional process of␣
 ↪synthesizing and testing materials is extremely slow and expensive. This␣
 ↪project demonstrates how a data-driven approach can accelerate this␣
 ↪discovery process, providing significant business value by reducing R&D␣
 ↪costs and shortening time-to-market.
#
# ## 3. Technical Skills & Achievements Demonstrated
#
# - **Structured Project Setup:** The project is organized with a modular `src`␣
 ↪directory, demonstrating best practices for creating reproducible and␣
 ↪maintainable code.
# - **Efficient Data Processing:** Raw chemical formulas are converted into a␣
 ↪rich set of over 100 physics-informed features. An efficient caching system␣
 ↪(`.parquet`) is implemented to dramatically speed up subsequent data loading␣
 ↪and processing.
```

```
# - **Insightful Visualization:** Professional, publication-quality␣
 ↪visualizations are used to analyze feature distributions and correlations.␣
 ↪The code demonstrates how to generate comprehensive, multi-page PDF reports␣
 ↪while keeping the notebook summary clean and concise.
# - **Statistical Rigor:** Visual observations are validated with formal␣
 ↪statistical tests (e.g., Shapiro-Wilk test for normality), confirming the␣
 ↪need for specific preprocessing steps like log transformations.
```

1. Environment Setup and Data Preparation

This section sets up the analysis environment. We import standard libraries and our custom modules from the `src` directory, which contains reusable functions for data processing, visualization, and environment configuration. We then load the raw data, generate a rich feature set from the chemical formulas, and cache the result as a `.parquet` file to accelerate future runs. This modular approach is a key practice for building reproducible and maintainable data science workflows.

[ ]:
```
import os
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
%matplotlib inline

# Ensure src directory is added to sys.path for modular imports
try:
    # Assumes the script is in the 'notebooks' directory
    PROJECT_ROOT = os.path.abspath(os.path.join(os.path.dirname(__file__), '..
 ↪'))
except NameError:
    # Fallback for interactive environments (Jupyter, VSCode)
    PROJECT_ROOT = os.path.abspath(os.path.join(os.getcwd()))

# Add the 'src' directory to the Python path
SRC_PATH = os.path.join(PROJECT_ROOT, 'src')
if SRC_PATH not in sys.path:
    sys.path.insert(0, SRC_PATH)

from utils import (
    load_or_process_dataframe,
    setup_environment,
    save_plot,
    style_df,
    log_and_print,
    perform_normality_tests
)
from viz import (
    plot_numeric_histograms_paginated,
```

```
    plot_numeric_histograms_log_paginated,
    plot_tc_histograms
)

# --- Setup Environment & Define Paths ---
setup_environment()
PLOTS_DIR = os.path.join(PROJECT_ROOT, 'plots', '1_eda')
os.makedirs(PLOTS_DIR, exist_ok=True)

CACHE_PATH = os.path.join(PROJECT_ROOT, 'data', 'processed', 'featurized.
 ↪parquet')
HIST_PATH = os.path.join(PLOTS_DIR, 'eda_numeric_histograms.pdf')
HIST_LOG_PATH = os.path.join(PLOTS_DIR, 'eda_numeric_histograms_log.pdf')
HIST_TC_PATH = os.path.join(PLOTS_DIR, 'eda_tc_histograms.pdf')
CORR_MATRIX_PATH = os.path.join(PLOTS_DIR, 'eda_corr_matrix.pdf')

# --- Load and Featurize Data (with Caching) ---
df = load_or_process_dataframe(cache_path=CACHE_PATH, project_root=PROJECT_ROOT)
log_and_print(f"Featurized dataframe shape: {df.shape}")
style_df(df.head())
```

```
No cache file found for c:\Users\angel\Thermal-Conductivity-
ML\data\processed\featurized.parquet (tried .parquet, .pkl, .csv)
Loading datasets…
Loading Citrine dataset…
Citrine dataset shape before dropping NaNs: (872, 5)
Citrine dataset shape after dropping NaNs: (872, 5)
Loading UCSB dataset…
UCSB dataset shape: (1093, 12)
Loading NIST dataset…
Reading NIST data file…
NIST dataset shape before filtering: (2689816, 70)
NIST dataset shape after filtering for crystalline solids: (176169, 70)
NIST dataset shape after standardizing property names: (1625, 70)
Citrine data loaded: (872, 6)
UCSB data loaded: (1093, 13)
NIST data loaded: (300, 4)
Shape after merging and cleaning: (757, 4)
Starting feature engineering…
Preparing composition features for 757 rows

ElementProperty:   0%|          | 0/757 [00:00<?, ?it/s]

Stoichiometry:   0%|          | 0/757 [00:00<?, ?it/s]

ValenceOrbital:   0%|          | 0/757 [00:00<?, ?it/s]

Composition features added.
Column names normalized.
Starting Materials Project query for 246 unique formulas…
```

```
Fetching Materials Project Data:   0%|          | 0/3 [00:00<?, ?it/s]
Retrieving SummaryDoc documents:   0%|          | 0/359 [00:00<?, ?it/s]
Fetching Materials Project Data:  33%|        | 1/3 [00:01<00:02,  1.31s/it]
Retrieving SummaryDoc documents:   0%|          | 0/559 [00:00<?, ?it/s]
Fetching Materials Project Data:  67%|       | 2/3 [00:02<00:01,  1.27s/it]
Retrieving SummaryDoc documents:   0%|          | 0/263 [00:00<?, ?it/s]
Fetching Materials Project Data: 100%|      | 3/3 [00:03<00:00,  1.26s/it]
Finished Materials Project query. Fetched data for 79 materials.
Materials Project features added.
Obtaining 3D dataset 76k …
Reference:https://www.nature.com/articles/s41524-020-00440-1
Other versions:https://doi.org/10.6084/m9.figshare.6815699
Loading the zipfile…
Loading completed.
Fetching JARVIS Data: 100%|      | 246/246 [00:00<00:00, 101615.01it/s]
Successfully fetched data for 70 of 246 unique formulas from JARVIS.


JARVIS features added.
Feature engineering complete.
Features cached to c:\Users\angel\Thermal-Conductivity-
ML\data\processed\featurized.parquet.
Processed and cached DataFrame to c:\Users\angel\Thermal-Conductivity-
ML\data\processed\featurized.parquet
Featurized dataframe shape: (757, 177)
```

[ ]:
```
<pandas.io.formats.style.Styler at 0x1bb8ea1ac10>
```

[ ]:
```python
# --- List All Feature Names ---
log_and_print("\n--- All Available Features ---")
# Convert to list and sort for easier reading
feature_list = sorted(df.columns.tolist())
for feature in feature_list:
    print(feature)
```

```
--- All Available Features ---
0-norm
10-norm
2-norm
3-norm
5-norm
7-norm
MagpieData_avg_dev_AtomicWeight
```

MagpieData_avg_dev_Column
MagpieData_avg_dev_CovalentRadius
MagpieData_avg_dev_Electronegativity
MagpieData_avg_dev_GSbandgap
MagpieData_avg_dev_GSmagmom
MagpieData_avg_dev_GSvolume_pa
MagpieData_avg_dev_MeltingT
MagpieData_avg_dev_MendeleevNumber
MagpieData_avg_dev_NUnfilled
MagpieData_avg_dev_NValence
MagpieData_avg_dev_NdUnfilled
MagpieData_avg_dev_NdValence
MagpieData_avg_dev_NfUnfilled
MagpieData_avg_dev_NfValence
MagpieData_avg_dev_NpUnfilled
MagpieData_avg_dev_NpValence
MagpieData_avg_dev_NsUnfilled
MagpieData_avg_dev_NsValence
MagpieData_avg_dev_Number
MagpieData_avg_dev_Row
MagpieData_avg_dev_SpaceGroupNumber
MagpieData_maximum_AtomicWeight
MagpieData_maximum_Column
MagpieData_maximum_CovalentRadius
MagpieData_maximum_Electronegativity
MagpieData_maximum_GSbandgap
MagpieData_maximum_GSmagmom
MagpieData_maximum_GSvolume_pa
MagpieData_maximum_MeltingT
MagpieData_maximum_MendeleevNumber
MagpieData_maximum_NUnfilled
MagpieData_maximum_NValence
MagpieData_maximum_NdUnfilled
MagpieData_maximum_NdValence
MagpieData_maximum_NfUnfilled
MagpieData_maximum_NfValence
MagpieData_maximum_NpUnfilled
MagpieData_maximum_NpValence
MagpieData_maximum_NsUnfilled
MagpieData_maximum_NsValence
MagpieData_maximum_Number
MagpieData_maximum_Row
MagpieData_maximum_SpaceGroupNumber
MagpieData_mean_AtomicWeight
MagpieData_mean_Column
MagpieData_mean_CovalentRadius
MagpieData_mean_Electronegativity
MagpieData_mean_GSbandgap

MagpieData_mean_GSmagmom
MagpieData_mean_GSvolume_pa
MagpieData_mean_MeltingT
MagpieData_mean_MendeleevNumber
MagpieData_mean_NUnfilled
MagpieData_mean_NValence
MagpieData_mean_NdUnfilled
MagpieData_mean_NdValence
MagpieData_mean_NfUnfilled
MagpieData_mean_NfValence
MagpieData_mean_NpUnfilled
MagpieData_mean_NpValence
MagpieData_mean_NsUnfilled
MagpieData_mean_NsValence
MagpieData_mean_Number
MagpieData_mean_Row
MagpieData_mean_SpaceGroupNumber
MagpieData_minimum_AtomicWeight
MagpieData_minimum_Column
MagpieData_minimum_CovalentRadius
MagpieData_minimum_Electronegativity
MagpieData_minimum_GSbandgap
MagpieData_minimum_GSmagmom
MagpieData_minimum_GSvolume_pa
MagpieData_minimum_MeltingT
MagpieData_minimum_MendeleevNumber
MagpieData_minimum_NUnfilled
MagpieData_minimum_NValence
MagpieData_minimum_NdUnfilled
MagpieData_minimum_NdValence
MagpieData_minimum_NfUnfilled
MagpieData_minimum_NfValence
MagpieData_minimum_NpUnfilled
MagpieData_minimum_NpValence
MagpieData_minimum_NsUnfilled
MagpieData_minimum_NsValence
MagpieData_minimum_Number
MagpieData_minimum_Row
MagpieData_minimum_SpaceGroupNumber
MagpieData_mode_AtomicWeight
MagpieData_mode_Column
MagpieData_mode_CovalentRadius
MagpieData_mode_Electronegativity
MagpieData_mode_GSbandgap
MagpieData_mode_GSmagmom
MagpieData_mode_GSvolume_pa
MagpieData_mode_MeltingT
MagpieData_mode_MendeleevNumber

MagpieData_mode_NUnfilled
MagpieData_mode_NValence
MagpieData_mode_NdUnfilled
MagpieData_mode_NdValence
MagpieData_mode_NfUnfilled
MagpieData_mode_NfValence
MagpieData_mode_NpUnfilled
MagpieData_mode_NpValence
MagpieData_mode_NsUnfilled
MagpieData_mode_NsValence
MagpieData_mode_Number
MagpieData_mode_Row
MagpieData_mode_SpaceGroupNumber
MagpieData_range_AtomicWeight
MagpieData_range_Column
MagpieData_range_CovalentRadius
MagpieData_range_Electronegativity
MagpieData_range_GSbandgap
MagpieData_range_GSmagmom
MagpieData_range_GSvolume_pa
MagpieData_range_MeltingT
MagpieData_range_MendeleevNumber
MagpieData_range_NUnfilled
MagpieData_range_NValence
MagpieData_range_NdUnfilled
MagpieData_range_NdValence
MagpieData_range_NfUnfilled
MagpieData_range_NfValence
MagpieData_range_NpUnfilled
MagpieData_range_NpValence
MagpieData_range_NsUnfilled
MagpieData_range_NsValence
MagpieData_range_Number
MagpieData_range_Row
MagpieData_range_SpaceGroupNumber
avg_d_valence_electrons
avg_f_valence_electrons
avg_p_valence_electrons
avg_s_valence_electrons
chemistry
crystal_structure
crystal_system
density
formula
frac_d_valence_electrons
frac_f_valence_electrons
frac_p_valence_electrons
frac_s_valence_electrons

```
jarvis_avg_bond_length
jarvis_avg_mass
jarvis_band_gap
jarvis_bulk_modulus
jarvis_debye_temp
jarvis_density
jarvis_elastic_anisotropy
jarvis_eps_electronic
jarvis_eps_total
jarvis_formation_energy
jarvis_is_metal
jarvis_min_bond_length
jarvis_natoms
jarvis_packing_fraction
jarvis_shear_modulus
material_id
mp_band_gap
mp_density
mp_energy_above_hull
mp_formula
mp_is_metal
mp_spacegroup
mp_volume
source
temperature
thermal_conductivity
```

2. Exploratory Data Analysis (EDA)

Feature Distributions

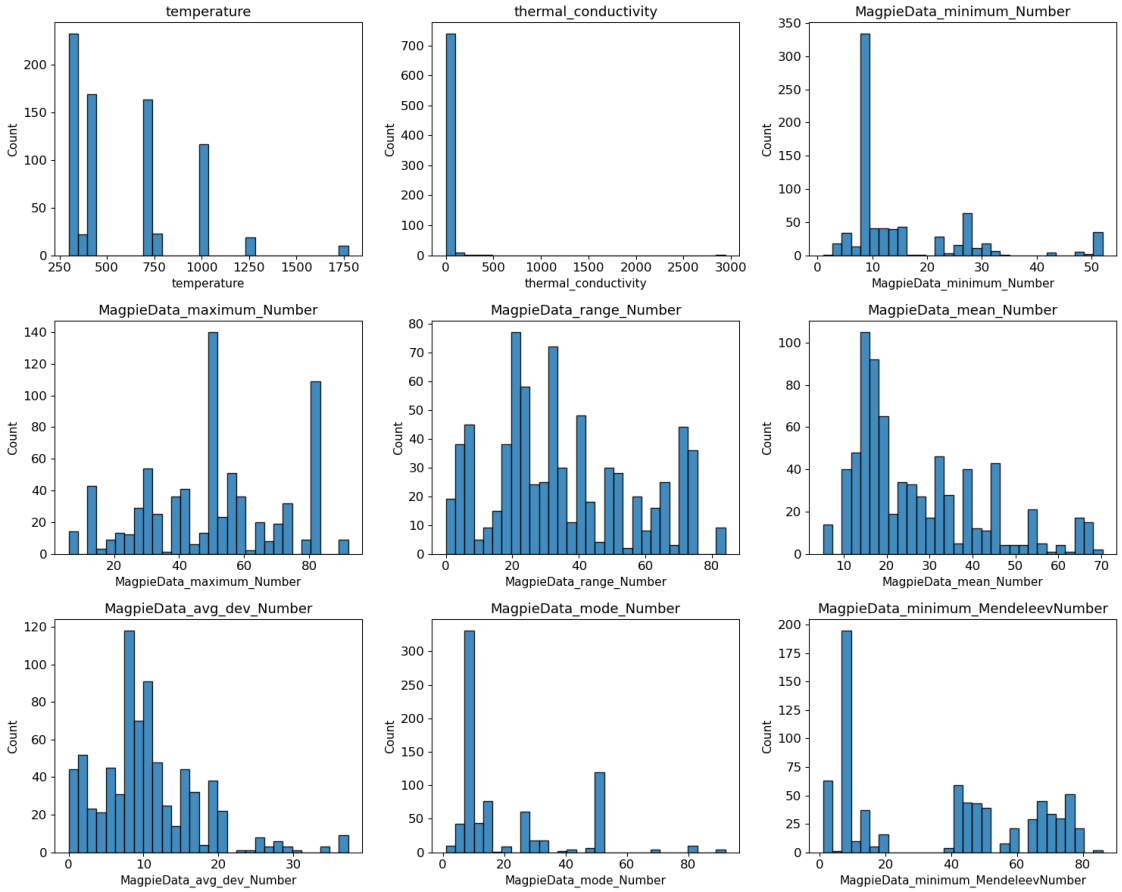
We begin by visualizing the distributions of the numeric features. This helps us understand the data's characteristics and identify potential issues such as skewness or outliers that could impact model performance.

To keep this notebook concise, we display only a **sample** of the histogram grids below (the first page of each set). The full set of histograms for all numeric features is saved as a multi-page PDF in the `plots/1_eda/` directory for detailed review.

[ ]:
```
# --- Generate and Save Multi-page PDFs for All Numeric Histograms ---

# Save paginated numeric histograms (linear scale)
figs_linear = plot_numeric_histograms_paginated(df, per_page=9)
with PdfPages(HIST_PATH) as pdf:
    for fig in figs_linear:
        pdf.savefig(fig)
        if fig.number > 1:
            plt.close(fig) # Close the figure after saving to avoid displaying␣
 ↪it inline
```

```
log_and_print(f"Paginated numeric histograms PDF saved to {HIST_PATH}")
```

```
Paginated numeric histograms PDF saved to c:\Users\angel\Thermal-Conductivity-
ML\plots\1_eda\eda_numeric_histograms.pdf
```

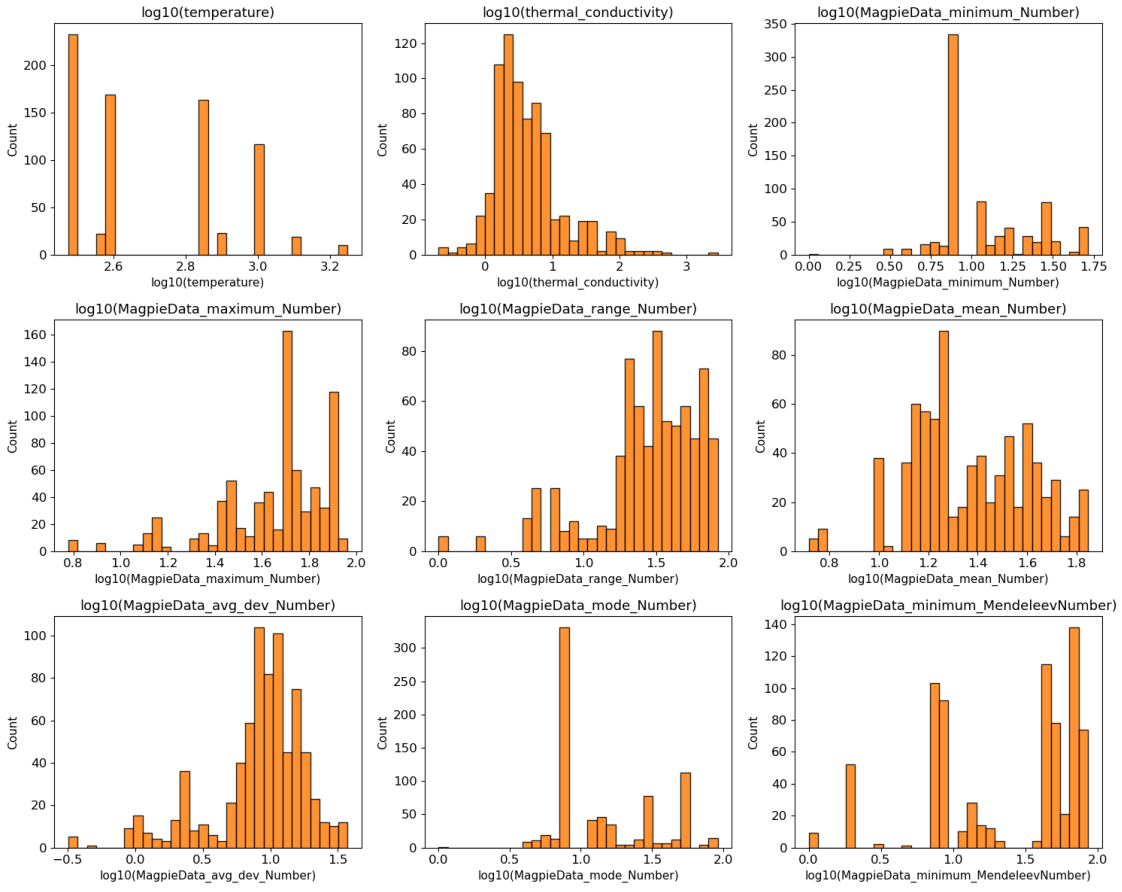


[ ]:
```
# Save paginated numeric histograms (log scale)
figs_log = plot_numeric_histograms_log_paginated(df, per_page=9)
with PdfPages(HIST_LOG_PATH) as pdf:
    for fig in figs_log:
        pdf.savefig(fig)
        if fig.number > 1:
            plt.close(fig) # Close the figure after saving to avoid displaying␣
 ↪it inline
log_and_print(f"Paginated log-scale numeric histograms PDF saved to␣
 ↪{HIST_LOG_PATH}")

# --- Display First Page Sample Inline ---
print("\nDisplaying a sample of the feature distributions (first page of 9␣
 ↪plots):")
```

```
figs_linear[0].suptitle('Sample of Numeric Feature Histograms (Linear Scale)',␣
  ↪fontsize=16, y=1.02)
plt.show()

print("\nDisplaying a sample of the log-transformed feature distributions:")
figs_log[0].suptitle('Sample of Numeric Feature Histograms (Log Scale)',␣
  ↪fontsize=16, y=1.02)
plt.show()
```

```
Paginated log-scale numeric histograms PDF saved to c:\Users\angel\Thermal-
Conductivity-ML\plots\1_eda\eda_numeric_histograms_log.pdf
```

```
Displaying a sample of the feature distributions (first page of 9 plots):
```




```
Displaying a sample of the log-transformed feature distributions:
```

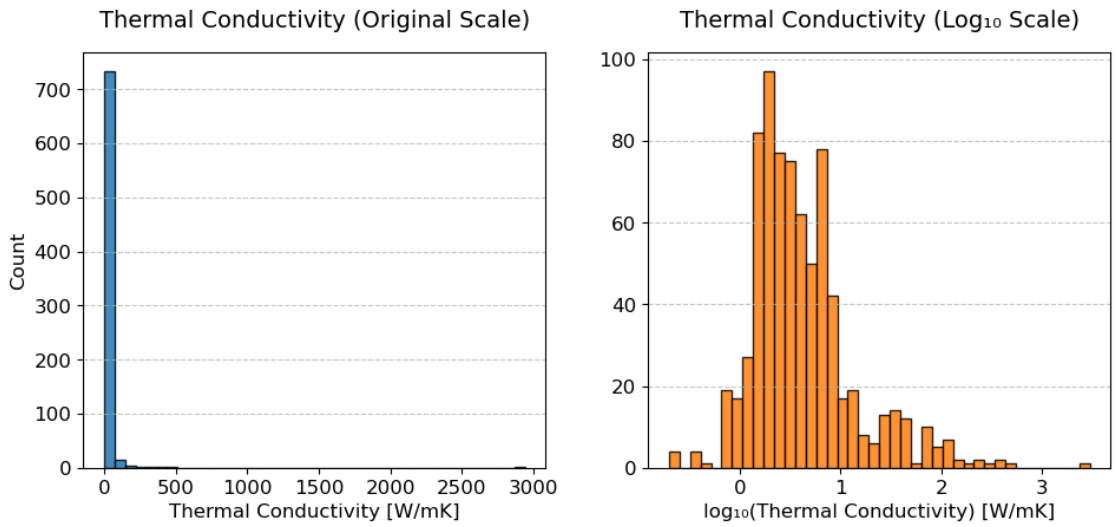
Target Variable: Thermal Conductivity

Now, let's focus on the target variable, `thermal_conductivity`. We'll visualize its distribution on both original and log scales. Assessing and correcting for skewness in the target is a critical step

for building robust predictive models.

[ ]:
```
# The plot_tc_histograms function from src/viz.py handles all styling.
fig_tc = plot_tc_histograms(df)
plt.savefig(HIST_TC_PATH, bbox_inches='tight')
log_and_print(f"Thermal conductivity histogram plot saved to {HIST_TC_PATH}")
plt.show()
```

```
Thermal conductivity histogram plot saved to c:\Users\angel\Thermal-
Conductivity-ML\plots\1_eda\eda_tc_histograms.pdf
```



Observations & Insights:

- **Right-Skewed Data:** The initial distributions show that many features, including the target variable `thermal_conductivity`, are heavily right-skewed. This is a common characteristic of physical property data.
- **Log Transformation:** Applying a log transformation (`log10`) to these skewed features results in distributions that are much closer to a symmetric, normal (Gaussian) distribution. This is particularly evident for `thermal_conductivity`, where the log-transformed version is more bell-shaped.
- **Modeling Implications:** Using these log-transformed features can lead to more stable and reliable performance in downstream modeling, especially for linear models, PCA, and other algorithms that benefit from normally distributed data. The side-by-side histograms clearly illustrate that the log-transformed target variable is a much better candidate for prediction.

Statistical Summary and Normality Assessment

To add statistical rigor to our visual analysis, we will now quantify our observations. This involves computing summary statistics and performing a formal normality test (Shapiro-Wilk) on the target variable, both before and after the log transformation.

[ ]:
```
# Summary statistics for thermal conductivity (original and log scale)
tc = df['thermal_conductivity'].dropna()
tc_log = pd.Series(np.log10(tc[tc > 0]), name='log10_thermal_conductivity')

summary_stats = {
    'Original': tc.describe(),
    'Log10': tc_log.describe()
}
summary_df = pd.DataFrame(summary_stats)
summary_df.loc['skew'] = [tc.skew(), tc_log.skew()]
summary_df.loc['kurtosis'] = [tc.kurtosis(), tc_log.kurtosis()]

log_and_print("Summary Statistics for Thermal Conductivity:")
style_df(summary_df)
```

```
Summary Statistics for Thermal Conductivity:
```

[ ]:
```
<pandas.io.formats.style.Styler at 0x1bb863dd450>
```

[ ]:
```
# Normality test (Shapiro-Wilk) for original and log-transformed target
from scipy.stats import shapiro
# Note: Shapiro-Wilk is reliable for n < 5000. We take a sample if data is␣
 ↪larger.
shapiro_orig = shapiro(tc.sample(min(len(tc), 5000), random_state=42))
shapiro_log = shapiro(tc_log.sample(min(len(tc_log), 5000), random_state=42))

log_and_print(f"\n--- Normality Test (Shapiro-Wilk) ---")
log_and_print(f"A p-value < 0.05 suggests the data is not normally distributed.
 ↪")
log_and_print(f"P-value (original): {shapiro_orig.pvalue:.3g}")
log_and_print(f"P-value (log10): {shapiro_log.pvalue:.3g}")
```

```
--- Normality Test (Shapiro-Wilk) ---
A p-value < 0.05 suggests the data is not normally distributed.
P-value (original): 5.78e-51
P-value (log10): 3.12e-20
```

[ ]:
```
# Perform normality tests on all numeric features
normality_results = perform_normality_tests(df)
log_and_print("\n--- Normality Tests for All Numeric Features ---")
style_df(normality_results.sort_values(by="P-Value", ascending=False))
```

```
--- Normality Tests for All Numeric Features ---

c:\Users\angel\.ai-navigator\micromamba\envs\cpu\Lib\site-
packages\scipy\stats\_axis_nan_policy.py:586: UserWarning:
```

```
scipy.stats.shapiro: Input data has range zero. The results may not be accurate.
```

[ ]:
```
<pandas.io.formats.style.Styler at 0x1bb863dd310>
```

Interpretation:

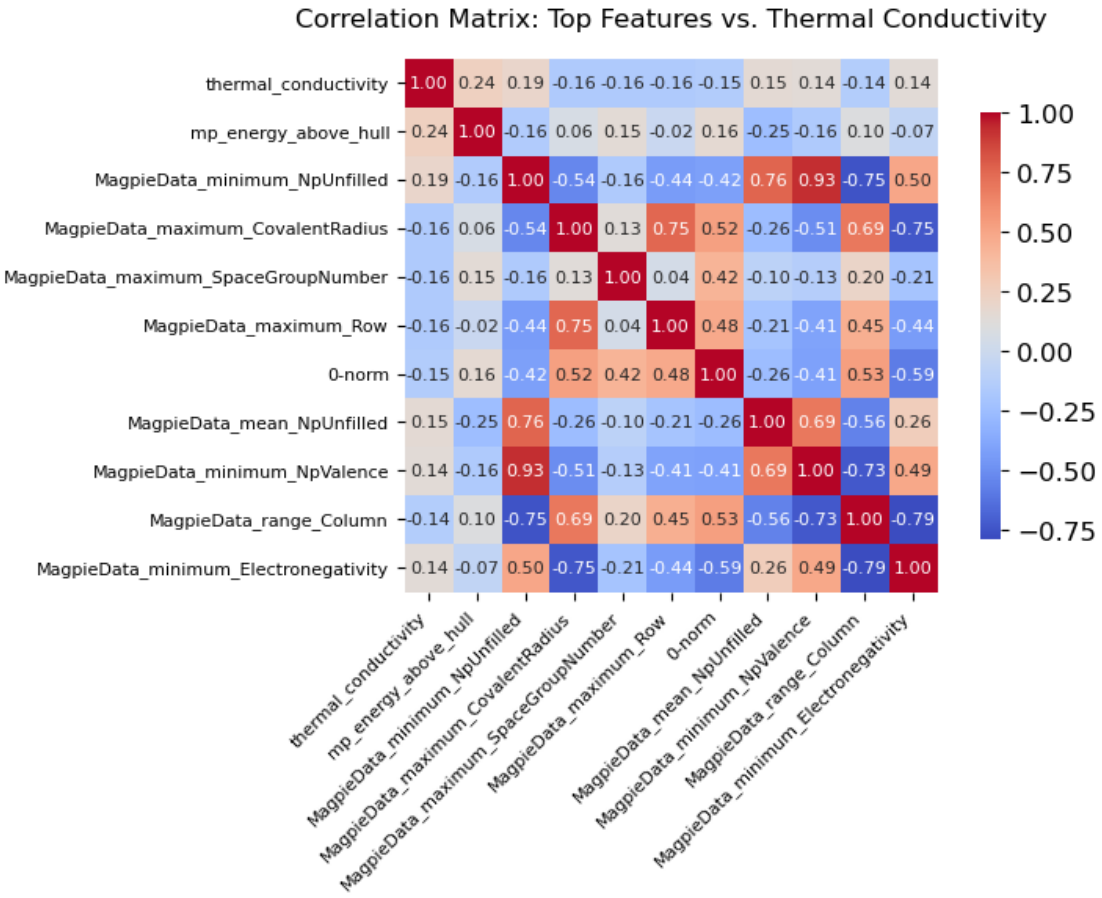
- The summary table confirms our visual assessment. The **skewness** drops from `23.8` to `1.27` after the log transformation, indicating a significant improvement in symmetry.
- The Shapiro-Wilk test p-values (where $p < 0.05$ suggests non-normality) provide statistical evidence of this improvement. While the log-transformed data is still not perfectly normal (p-value is very small), it is substantially closer and better satisfies the assumptions of many models.
- The table above shows the normality test results for all numeric features. Most features are not normally distributed, which reinforces the importance of using transformations or non-parametric methods in subsequent modeling steps.

Correlation Analysis

Finally, we examine the correlation structure of the dataset. A correlation matrix helps us identify which features are most strongly related to the target variable (`thermal_conductivity`). It also reveals potential multicollinearity (high correlation between predictor variables), an important consideration for feature selection in the modeling phase.

[ ]:
```
# Compute correlation matrix for top 10 features most correlated with the target
top_corr_features = df.corr(numeric_only=True)['thermal_conductivity'].abs().
 ↪sort_values(ascending=False).head(11).index
corr_matrix = df[top_corr_features].corr()

import seaborn as sns
plt.figure(figsize=(8, 6))
# Use smaller font for annotations and axis labels for readability
ax = sns.heatmap(
    corr_matrix,
    annot=True,
    fmt='.2f',
    cmap='coolwarm',
    square=True,
    annot_kws={"size": 8},
    cbar_kws={"shrink": 0.8}
)
plt.title('Correlation Matrix: Top Features vs. Thermal Conductivity',␣
 ↪fontsize=12, pad=15)
plt.xticks(fontsize=8, rotation=45, ha='right')
plt.yticks(fontsize=8, rotation=0)
plt.tight_layout()
plt.savefig(CORR_MATRIX_PATH, bbox_inches='tight')
log_and_print(f"Correlation matrix plot saved to {CORR_MATRIX_PATH}")
plt.show()
```

```
Correlation matrix plot saved to c:\Users\angel\Thermal-Conductivity-
ML\plots\1_eda\eda_corr_matrix.pdf
```



Interpretation:

- The heatmap above highlights the features most strongly correlated with thermal conductivity, such as `energy above hull` and `minimum_NpUnfilled`.
- We also observe high correlations between some independent features (e.g., `minimum_NpUnfilled` and `minimum_NpValence`). This indicates potential multicollinearity, which can affect the interpretability and stability of some linear models. This analysis is crucial for guiding feature selection and engineering in the subsequent modeling stages.

3. Conclusion and Next Steps

This exploratory analysis has provided critical insights into the dataset's structure. We've confirmed the necessity of log-transforming our skewed target variable and identified key feature correlations that will inform our modeling strategy. The data is now understood and prepared for the next stage of the pipeline.

The workflow continues in the next notebook, where we will use unsupervised learning techniques to uncover hidden structures in the data:

Next Notebook: `2_clustering_and_pca.py`