Exploratory Data Analysis

The Table in BigQuery contains 21,264 rows. In the interest of speed and rapid iteration, we will not operate on all the rows of this dataset, but rather, we will select a thousand rows for data exploration, transformation, and machine learning spot checking.

```
import pandas as pd
%%bigquery --project ekabasandbox super_cond_df
WITH super_df AS (
SELECT
  number_of_elements, mean_atomic_mass, wtd_mean_atomic_mass,
  gmean_atomic_mass, wtd_gmean_atomic_mass, entropy_atomic_mass,
  wtd_entropy_atomic_mass, range_atomic_mass, wtd_range_atomic_mass,
  std_atomic_mass, wtd_std_atomic_mass, mean_fie, wtd_mean_fie,
  gmean_fie, wtd_gmean_fie, entropy_fie, wtd_entropy_fie, range_fie,
  wtd_range_fie, std_fie, wtd_std_fie, mean_atomic_radius, wtd_mean_atomic_
  radius,
  gmean_atomic_radius, wtd_gmean_atomic_radius, entropy_atomic_radius,
  wtd_entropy_atomic_radius, range_atomic_radius, wtd_range_atomic_radius,
  std_atomic_radius, wtd_std_atomic_radius, mean_Density, wtd_mean_Density,
  gmean_Density, wtd_gmean_Density, entropy_Density, wtd_entropy_Density,
  range_Density, wtd_range_Density, std_Density, wtd_std_Density, mean_
  ElectronAffinity,
  wtd_mean_ElectronAffinity, gmean_ElectronAffinity, wtd_gmean_
  ElectronAffinity
  entropy_ElectronAffinity, wtd_entropy_ElectronAffinity, range_
  ElectronAffinity,
  wtd_range_ElectronAffinity, std_ElectronAffinity, wtd_std_
  ElectronAffinity,
  mean_FusionHeat, wtd_mean_FusionHeat, gmean_FusionHeat, wtd_gmean_
  FusionHeat,
  entropy_FusionHeat, wtd_entropy_FusionHeat, range_FusionHeat,
  wtd_range_FusionHeat, std_FusionHeat, wtd_std_FusionHeat, mean_
  ThermalConductivity,
  wtd_mean_ThermalConductivity, gmean_ThermalConductivity, wtd_gmean_
  ThermalConductivity,
  entropy_ThermalConductivity, wtd_entropy_ThermalConductivity, range_
  ThermalConductivity,
  wtd_range_ThermalConductivity, std_ThermalConductivity, wtd_std_
  ThermalConductivity,
  mean_Valence, wtd_mean_Valence, gmean_Valence, wtd_gmean_Valence,
  entropy_Valence, wtd_entropy_Valence, range_Valence, wtd_range_Valence,
  std_Valence, wtd_std_Valence, critical_temp, ROW_NUMBER() OVER (PARTITION
  BY number_of_elements) row_num
FROM
  `superconductor.superconductor` )

SELECT
  *
FROM
  super_df
LIMIT
  1000
# Dataframe shape
super_cond_df.shape
```

Next, we'll explore the dataset to gain more understanding of the features and their relationships. This process is called exploratory data analysis (EDA).

- Check the column datatypes.

```
# check column datatypes
super_cond_df.dtypes
```

```
number_of_elements                int64
mean_atomic_mass                float64
wtd_mean_atomic_mass            float64
gmean_atomic_mass               float64
wtd_gmean_atomic_mass           float64
entropy_atomic_mass             float64
wtd_entropy_atomic_mass         float64
                                  ...
range_Valence                     int64
wtd_range_Valence               float64
```

```
std_Valence                        float64
wtd_std_Valence                    float64
critical_temp                      float64
row_num                              int64
Length: 82, dtype: object
```

From the results, all the data attributes are of numeric type:

- Next, we will use a tool called **pandas profiling**. This package produces a full range of exploratory data analytics for a Pandas DataFrame object. The result includes summary statistics of the dataset such as the number of variables, number of data observations, and number of missing values (if any). It also includes a histogram visualization for each attribute, descriptive statistics (such as the mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, and skewness), and quantile statistics (such as minimum value, Q1, median, Q3, maximum, range, and interquartile range). Also, the profile produces multivariate correlation graphs and produces a list of variables that are highly correlated.

  Import the pandas profiling library.

  ```
  # pandas profiling
  import pandas_profiling
  ```

  Run the profile and save the output.

  ```
  # run report
  profile_result = pandas_profiling.ProfileReport(super_cond_df)
  ```

To view the complete report, run the saved output variable:

```
profile_result
```

- Retrieve the rejected variables (i.e, attributes with high correlation).

  ```
  # get rejected variables (i.e, attributes with high correlation)
  rejected_vars = profile_result.get_rejected_variables
  ```

- Filter the dataset columns by removing the variables with high correlation.

```
# filter from attributes set
super_cond_df.drop(rejected_vars(), axis=1, inplace=True)
```

- Next, standardize the dataset values so that they fall within the same scale range (we'll be using Scikit-learn minmax_scale function). Standardizing the values improves the predictive performance of the model because the optimization algorithm can better minimize the cost function.

```
# scale the dataframe values
from sklearn.preprocessing import minmax_scale

dataset = pd.DataFrame(minmax_scale(super_cond_df), columns=
super_cond_df.columns)
```

- Also, the attribute values are normalized so that the distribution more closely resembles a normal or Gaussian distribution. This technique is also noted to have a positive impact on the model performance.

```
# normalize the dataframe
from sklearn.preprocessing import Normalizer

dataset = pd.DataFrame(Normalizer().fit(dataset).
transform(dataset), columns=dataset.columns)
```

- Plot the histogram distribution of the variables (see Figure 44-2).

```
# plot the histogram distribution of the variables
import matplotlib.pyplot as plt

%matplotlib inline

dataset.hist(figsize=(18, 18))
plt.show()
```

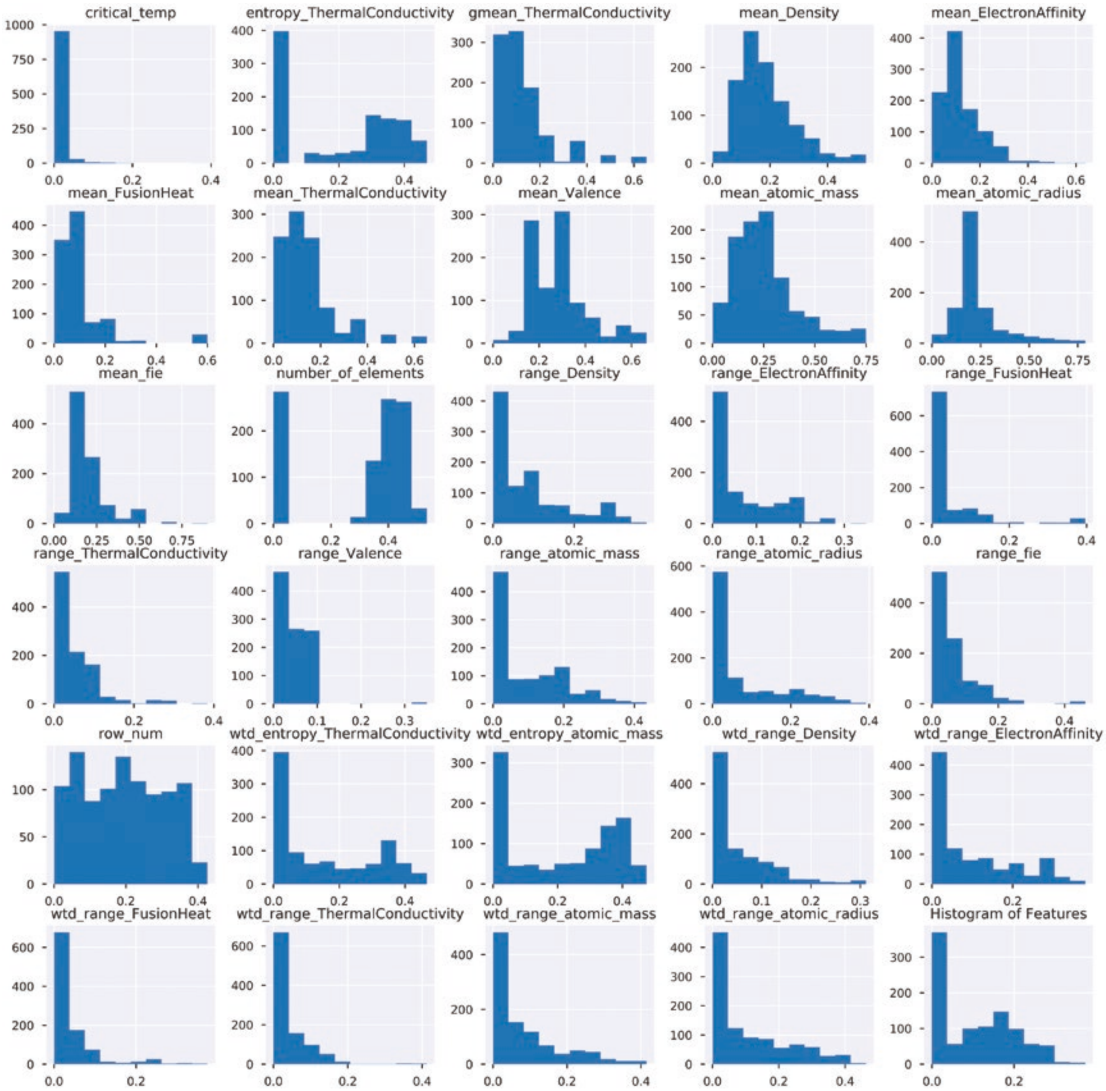

Figure 44-2. *Histogram showing variable distribution*

Spot Checking Machine Learning Algorithms

With our reduced dataset, let’s sample a few candidate algorithms to have an idea on their performance and which is more likely to work best for this problem domain. Let’s take the following steps:

- The dataset is split into a design matrix and their corresponding label vector.