Case Study 1

Your case study is to build a linear regression model using L1 or L2 regularization (or both) the task to predict the Critical Temperature as closely as possible. In addition, include in your write-up which variable carries the most importance.

In [ ]:
```
# General libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# sklearn libraries
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.linear_model import LinearRegression, Lasso, Ridge
import sklearn.metrics as metrics
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import make_pipeline
```

In [ ]:
```
# Read in the data
unique = pd.read_csv('unique_m.csv')
train = pd.read_csv('train.csv')

# Drop critical temp since it exists in both data frames
unique = unique.drop(['critical_temp'], axis = 1)

# Merge unique and train
df = pd.concat([train, unique], axis = 1)
```

Examine the Data

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21263 entries, 0 to 21262
Columns: 169 entries, number_of_elements to material
dtypes: float64(156), int64(12), object(1)
memory usage: 27.4+ MB
```

In [ ]:
```
df.head()
```

Out[ ]:

	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomic_mass
0	4	88.944468	57.862692	66.361592
1	5	92.729214	58.518416	73.132787
2	4	88.944468	57.885242	66.361592
3	4	88.944468	57.873967	66.361592
4	4	88.944468	57.840143	66.361592

5 rows × 169 columns

In [ ]:
```
df.describe()
```

Out[ ]:

	number_of_elements	mean_atomic_mass	wtd_mean_atomic_mass	gmean_atomic_ma
count	21263.000000	21263.000000	21263.000000	21263.0000
mean	4.115224	87.557631	72.988310	71.2906
std	1.439295	29.676497	33.490406	31.0302
min	1.000000	6.941000	6.423452	5.3205
25%	3.000000	72.458076	52.143839	58.0412
50%	4.000000	84.922750	60.696571	66.3615
75%	5.000000	100.404410	86.103540	78.1166
max	9.000000	208.980400	208.980400	208.9804

8 rows × 168 columns

Material isn't listed in `df.describe()`. How many unique values does it have?

In [ ]:
```
len(df['material'].unique())
```

Out[ ]:
```
15542
```

Yikes, 15,542 unique strings. I'll drop this column from the data.

In [ ]:
```
df=df.drop(['material'], axis=1)
```

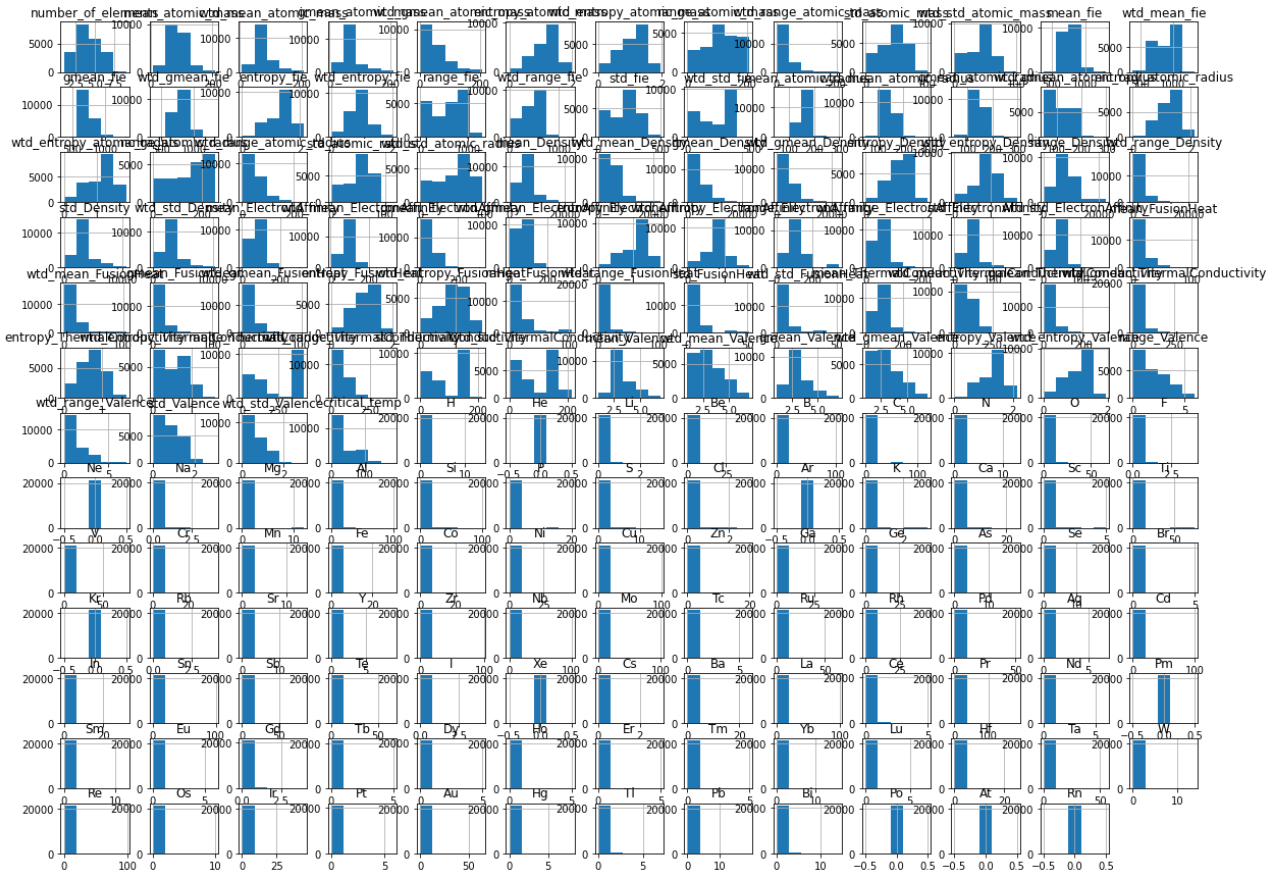
Missing Values

In [ ]:
```
df.isnull().sum()
```

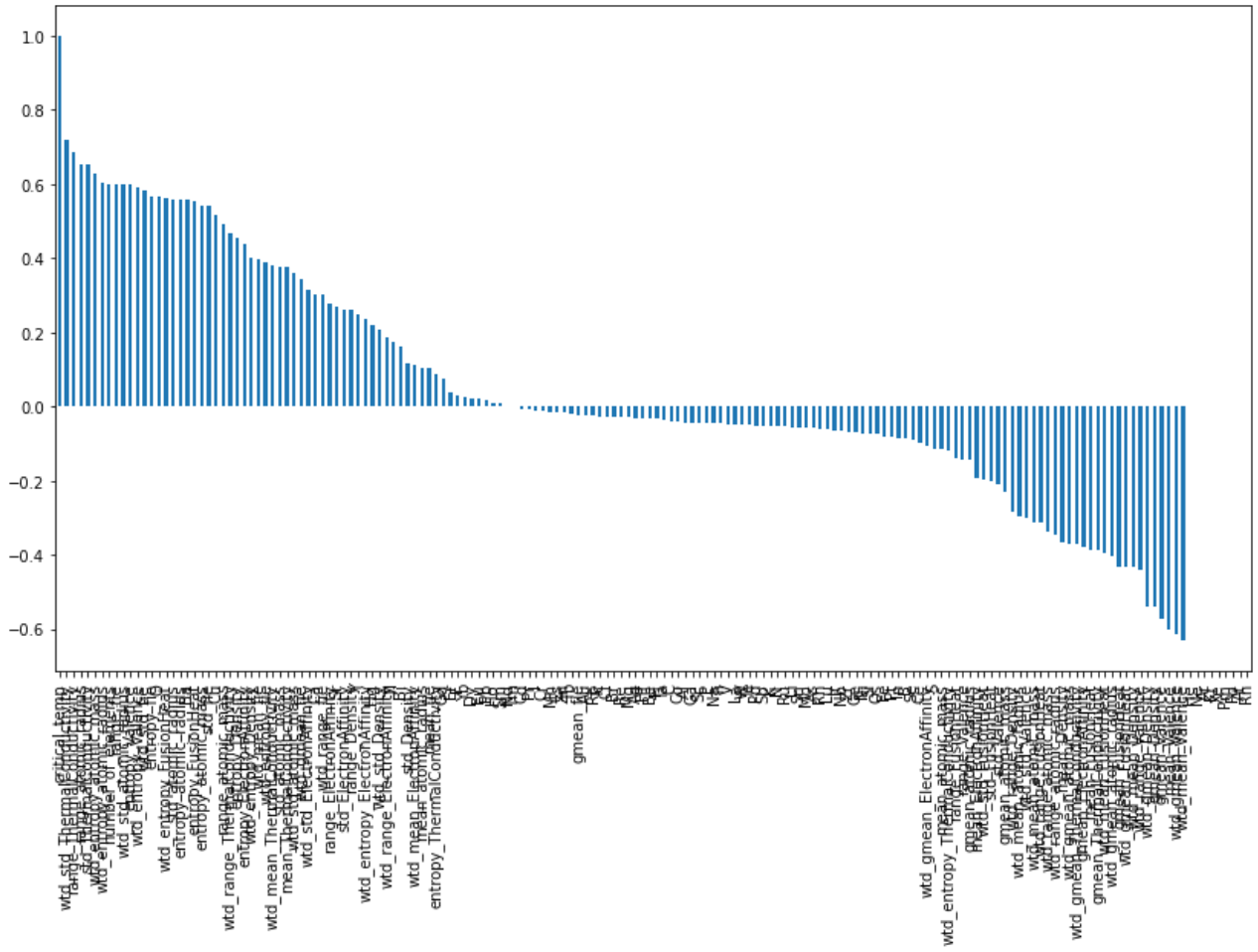
Out[ ]:
```
number_of_elements        0
mean_atomic_mass          0
wtd_mean_atomic_mass      0
gmean_atomic_mass         0
wtd_gmean_atomic_mass     0
                         ..
Pb                        0
Bi                        0
Po                        0
At                        0
Rn                        0
Length: 168, dtype: int64
```

It doesn't appear that there are any missing values in the data.

In [ ]:
```
df.hist(bins=5, figsize=(20,15))
plt.show()
```



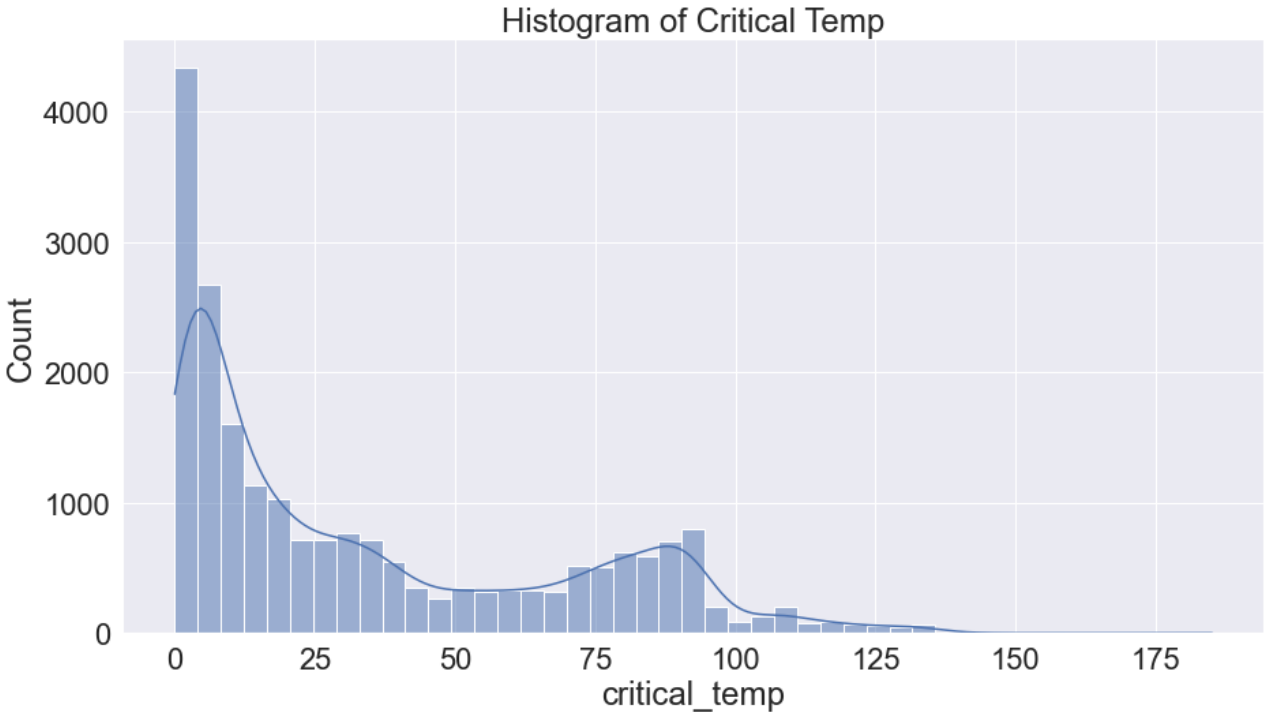
In [ ]:
```
plt.figure(figsize=(14,8))
bars = df.corr()['critical_temp'].sort_values(ascending=False).plot(kind='ba
```



Examine Response Variable

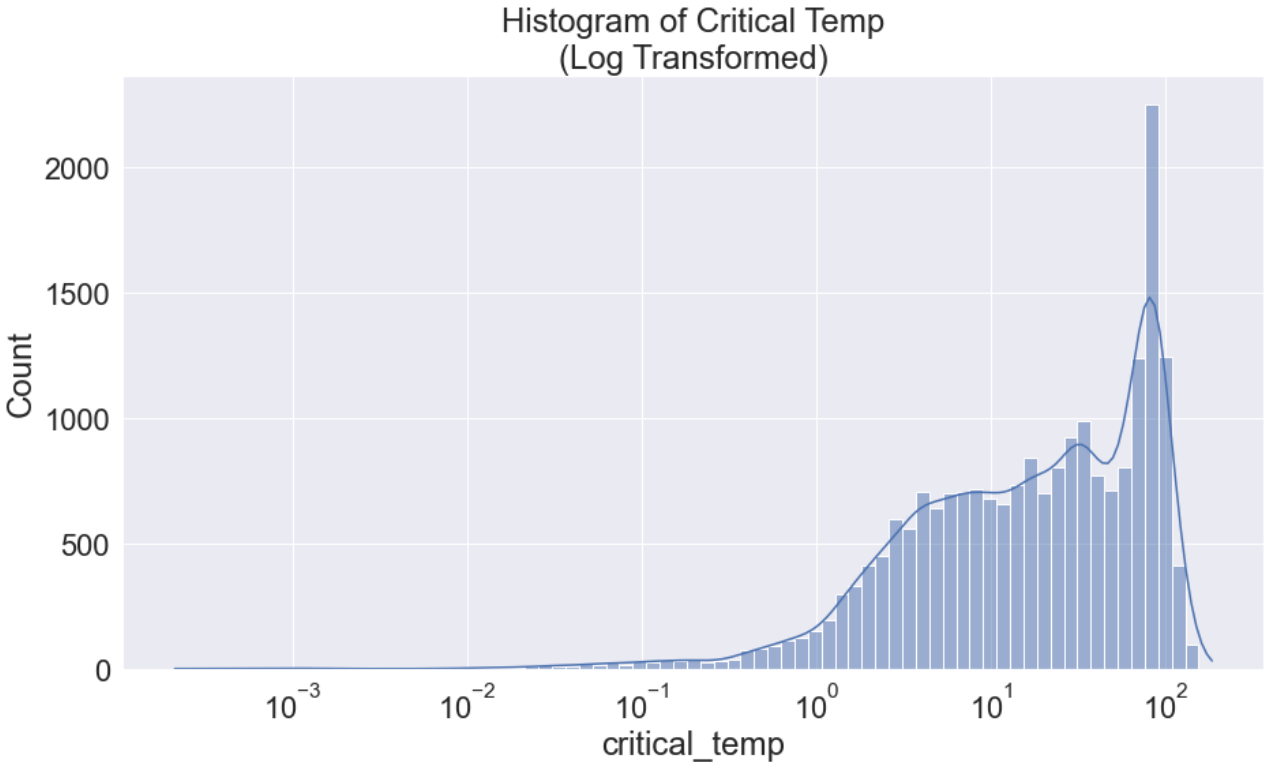
In [ ]:
```
# Examine response variable
sns.set(rc = {'figure.figsize':(15,8)})
sns.set(font_scale = 2)
p=sns.histplot(df['critical_temp'], kde=True)
p.set_title("Histogram of Critical Temp")
```

Out[ ]:
```
Text(0.5, 1.0, 'Histogram of Critical Temp')
```



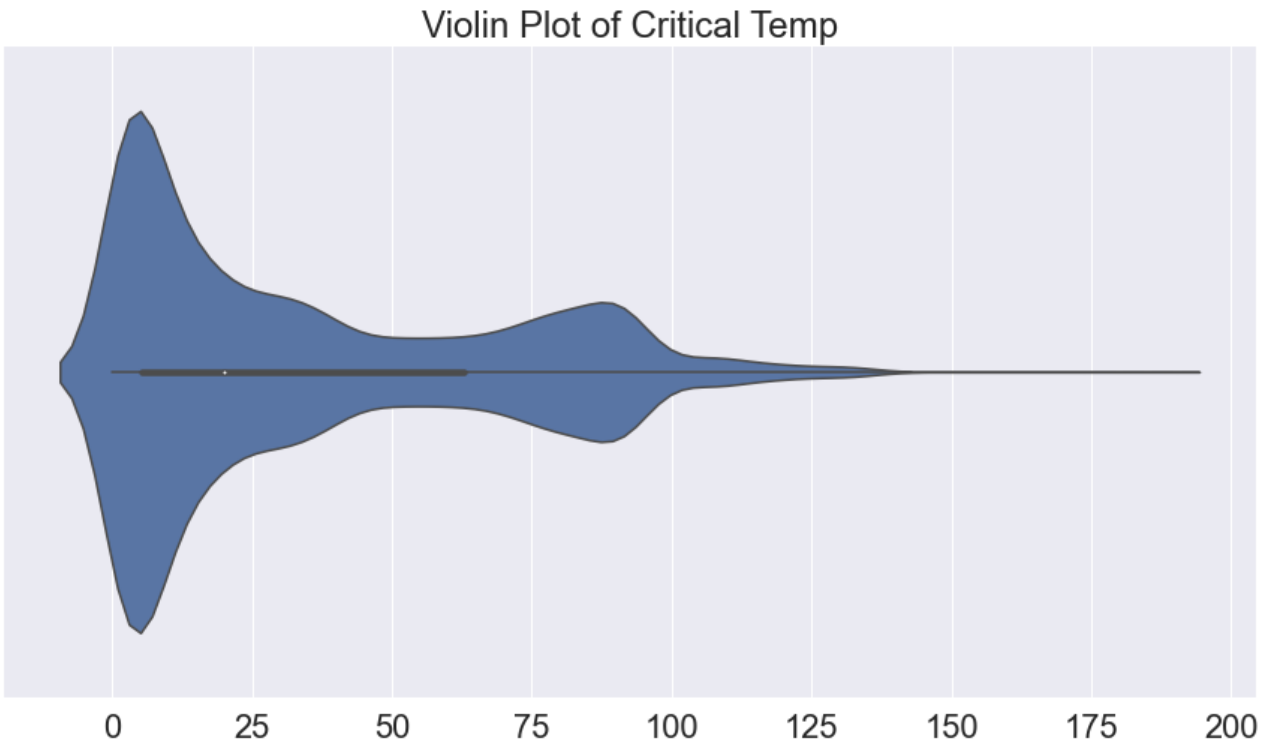
In [ ]:
```
# Does a transformation help?
p=sns.histplot(df['critical_temp'], kde=True, log_scale=True)
p.set_title("Histogram of Critical Temp\n(Log Transformed)")
```

Out[ ]:
```
Text(0.5, 1.0, 'Histogram of Critical Temp\n(Log Transformed)')
```



In [ ]:
```
p=sns.violinplot(x=df['critical_temp'])
p.set_title("Violin Plot of Critical Temp")
p.set(xlabel=None)
```

Out[ ]:
```
[Text(0.5, 0, '')]
```



Model Building

Define & Split the Data

In [ ]:
```
# Define response & feature variables
X = df.drop(labels = ['critical_temp'], axis = 1)
y = df['critical_temp']

# Create a feature list
feature_list = list(X.columns)

# Split the data
X_train, X_test, y_train, y_test =\
    train_test_split(X, y,
    test_size=0.25,
    random_state=1)
```

Instantiate Pipeline

In [ ]:
```
# Source: https://towardsdatascience.com/pre-process-data-with-pipeline-to-p

lasso_pipeline = make_pipeline(RobustScaler(), Lasso(random_state=1))
ridge_pipeline = make_pipeline(RobustScaler(), Ridge(random_state=1))

# Set up alpha search
# https://stackoverflow.com/questions/41899132/invalid-parameter-for-sklearn
alpha_range = [0.0001, 0.001, 0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 1, 3, 5, 7, 9,
lasso_params = [{'lasso__alpha': alpha_range}]
ridge_params = [{'ridge__alpha': alpha_range}]
```

Train Models

LASSO

In [ ]:
```
lasso_train = GridSearchCV(estimator=lasso_pipeline,
                           param_grid=lasso_params,
                           scoring='neg_mean_absolute_error',
                           cv=10,
                           n_jobs=-1)

lasso_train.fit(X_train, y_train)

print("L1 (LASSO) Model")
print("Best Score:", lasso_train.best_score_)
print("Best Alpha:", lasso_train.best_params_)
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.956e+06, tolerance: 1.698e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.919e+06, tolerance: 1.696e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.950e+06, tolerance: 1.690e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.964e+06, tolerance: 1.700e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.957e+06, tolerance: 1.700e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.679e+06, tolerance: 1.698e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
```

```
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.650e+06, tolerance: 1.695e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.922e+06, tolerance: 1.695e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.650e+06, tolerance: 1.696e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.952e+06, tolerance: 1.695e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.713e+06, tolerance: 1.700e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.946e+06, tolerance: 1.687e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.663e+06, tolerance: 1.700e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.687e+06, tolerance: 1.695e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.951e+06, tolerance: 1.698e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.936e+06, tolerance: 1.694e+03
```

```
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 2.879e+05, tolerance: 1.698e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 2.984e+05, tolerance: 1.695e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 2.929e+05, tolerance: 1.700e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 3.236e+05, tolerance: 1.695e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 4.497e+05, tolerance: 1.696e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 2.971e+05, tolerance: 1.700e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 3.147e+05, tolerance: 1.690e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.700e+06, tolerance: 1.690e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 2.883e+05, tolerance: 1.687e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
```

```
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 2.975e+05, tolerance: 1.698e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 3.667e+05, tolerance: 1.694e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.675e+06, tolerance: 1.698e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.716e+06, tolerance: 1.694e+03
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.686e+06, tolerance: 1.687e+03
  model = cd_fast.enet_coordinate_descent(
L1 (LASSO) Model
Best Score: -12.675935647682858
Best Alpha: {'lasso__alpha': 0.001}
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:647: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iterations
, check the scale of the features or consider increasing regularisation. Dua
lity gap: 1.869e+06, tolerance: 1.884e+03
  model = cd_fast.enet_coordinate_descent(
```

Ridge

In [ ]:
```
ridge_train = GridSearchCV(estimator=ridge_pipeline,
                           param_grid=ridge_params,
                           scoring='neg_mean_absolute_error',
                           cv=10,
                           n_jobs=-1)

ridge_train.fit(X_train, y_train)

print("L2 (Ridge) Model")
print("Best Score:", ridge_train.best_score_)
print("Best Alpha:", ridge_train.best_params_)
```

```
L2 (Ridge) Model
Best Score: -12.635863718438099
Best Alpha: {'ridge__alpha': 0.3}
```

Test Models

In [ ]:
```
y_lasso_pred = lasso_train.predict(X_test)
y_ridge_pred = ridge_train.predict(X_test)

print("L1 (Lasso) Performance")
print("R2:", metrics.r2_score(y_test, y_lasso_pred))
print("MAE:", metrics.mean_absolute_error(y_test, y_lasso_pred))

print("L2 (Ridge) Performance")
print("R2:", metrics.r2_score(y_test, y_ridge_pred))
print("MAE:", metrics.mean_absolute_error(y_test, y_ridge_pred))
```

```
L1 (Lasso) Performance
R2: 0.4747144301704468
MAE: 12.926775749215134
L2 (Ridge) Performance
R2: 0.4736246385198172
MAE: 12.916091908191113
```

LASSO Coefficients

In [ ]:
```
lasso_train.best_estimator_['lasso'].coef_
```

Out[ ]:
```
array([ 9.57783500e-01,  1.07102790e+01, -2.44358911e+01,  1.04027772e+00,
        8.16278998e+00, -1.53256296e+01,  6.87766444e+00,  1.35714866e+01,
        2.76912117e+00, -7.29225020e+00, -4.71937808e+00, -5.00386011e+00,
        1.11240793e+01,  3.03823513e+00, -1.47706986e-01,  2.79798655e+01,
        1.24165866e+01,  2.51236245e+01,  7.15162495e+00, -2.23953985e+01,
       -9.81364123e-01,  4.40499564e+00,  2.28749791e+01, -1.06063075e+01,
       -1.36852157e+01, -1.87685238e+01,  2.81080137e+00,  1.15680374e+01,
       -2.62937064e+00, -2.14754443e+01,  1.42423094e+01, -8.90430382e+00,
        1.45265338e+01, -2.53589478e+00,  9.48326449e-01, -1.65980781e+00,
       -1.92615392e+00, -2.79048700e+00, -1.49186569e+00,  3.00868899e+00,
       -7.40083286e-01, -2.21373455e+00,  1.65227462e+01,  5.16101130e+00,
       -2.01803338e+01,  2.33521312e+00, -3.17175823e+00, -1.81894675e+01,
       -3.47078488e+00,  2.12717758e+01, -1.05784925e+01,  4.41433965e+00,
       -1.46848711e+01, -7.57258539e-01,  9.28403492e+00, -7.06984796e+00,
        1.08400838e+01, -2.75456654e+00,  4.50022333e+00, -3.74596540e-01,
        1.22957938e+00, -1.06763558e+00,  2.03920295e+01, -3.32826257e+00,
       -1.35831327e+01,  4.36822098e+00,  1.78870150e+00, -3.04940941e+01,
       -1.10573467e+01,  2.92823232e+01,  5.16633945e+00, -3.70894323e+00,
       -2.63565941e+00,  6.33971955e+00,  3.74889889e-01,  9.13386712e+00,
       -2.91985086e+01,  1.22043044e+01, -1.39216655e+00, -3.48766522e+00,
       -1.20790538e+01, -1.59036674e+00,  0.00000000e+00,  2.03980073e+00,
       -2.07088140e-01, -8.96282716e-01, -5.37536627e-02, -8.57234841e-01,
       -2.85888644e+00,  7.85478426e+00,  0.00000000e+00,  4.93087301e+00,
        1.92302465e-01, -1.66773375e-01, -1.63523818e+00, -1.37116598e+00,
       -1.61602981e+00, -6.57462661e+00,  0.00000000e+00,  7.91926303e+00,
        1.52521853e+00,  2.77876592e-01, -6.33380191e-02,  2.70928383e-02,
       -3.40225311e-01, -6.57350325e-01,  1.12988964e+00, -4.90142912e-01,
       -3.90928251e-01, -1.62054044e+00, -6.77624016e-02,  4.15975316e-01,
       -1.03629972e+00, -1.73660252e+00, -9.61018888e-01,  4.49519403e-01,
        0.00000000e+00,  7.99866829e+00,  1.59816293e-01, -2.40118167e-01,
        4.58537181e-03,  8.97130750e-02,  1.12198379e-01,  1.81761821e+00,
        2.69863413e-01, -5.06208729e-02,  4.71838080e-02, -1.07904524e+01,
       -1.30990690e+01,  1.29731285e+00, -4.75534544e-02, -2.09319119e-01,
        6.42825888e-01,  6.97522861e+00,  0.00000000e+00,  5.59681256e+00,
        1.18779051e+01, -3.68035330e-02, -2.36269273e+00, -4.92670364e-02,
       -2.25105371e+00,  0.00000000e+00,  2.99732327e-01, -1.77472869e+00,
       -6.45399400e-01,  1.72880767e+00,  5.38477121e+00,  3.05348872e+00,
        2.29729229e+00,  0.00000000e+00,  2.72754219e+00,  3.89535507e+00,
       -5.56428177e-01, -6.56803226e-02,  2.22277354e-01, -9.26692325e-02,
        1.34390256e+00,  9.35943109e-02,  4.24995960e+00, -5.18340417e-01,
        5.44217393e+00,  5.02505209e+00,  1.82571149e+00,  5.44537540e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00])
```

In [ ]:
```
lasso_weights = {df.columns[key]:abs(value) for key, value in enumerate(lass
dict(sorted(lasso_weights.items(), key=lambda item: item[1], reverse=True))
```

Out[ ]:
```
{'range_ThermalConductivity': 30.49409411677583,
 'std_ThermalConductivity': 29.282323240875897,
 'wtd_entropy_Valence': 29.198508585454952,
 'entropy_fie': 27.97986546194415,
 'range_fie': 25.123624476195126,
 'wtd_mean_atomic_mass': 24.435891139005925,
 'wtd_mean_atomic_radius': 22.874979120467305,
 'std_fie': 22.39539852273098,
 'std_atomic_radius': 21.475444256410604,
 'std_ElectronAffinity': 21.27177577566263,
 'wtd_mean_ThermalConductivity': 20.392029472833922,
```

```
'wtd_gmean_ElectronAffinity': 20.180333799299305,
'entropy_atomic_radius': 18.76852382229849,
'range_ElectronAffinity': 18.189467525878282,
'wtd_mean_ElectronAffinity': 16.52274622173433,
'entropy_atomic_mass': 15.325629639254325,
'wtd_mean_FusionHeat': 14.6848711033669,
'wtd_mean_Density': 14.52653380481855,
'wtd_std_atomic_radius': 14.242309400501746,
'wtd_gmean_atomic_radius': 13.685215742990312,
'wtd_gmean_ThermalConductivity': 13.583132665970748,
'range_atomic_mass': 13.571486647086104,
'Ag': 13.099068959396925,
'wtd_entropy_fie': 12.416586573489242,
'range_Valence': 12.204304386174638,
'wtd_std_Valence': 12.07905375876434,
'Cs': 11.877905091518043,
'range_atomic_radius': 11.568037372445035,
'wtd_mean_fie': 11.124079338792933,
'wtd_range_ThermalConductivity': 11.057346695129356,
'wtd_entropy_FusionHeat': 10.840083795372646,
'Pd': 10.79045244425821,
'mean_atomic_mass': 10.710278983793811,
'gmean_atomic_radius': 10.606307488553403,
'wtd_std_ElectronAffinity': 10.578492540726353,
'wtd_gmean_FusionHeat': 9.284034916446018,
'entropy_Valence': 9.13386711623424,
'mean_Density': 8.904303815908566,
'wtd_gmean_atomic_mass': 8.162789976403497,
'Kr': 7.998668286799644,
'Ar': 7.919263030464211,
'O': 7.854784255642465,
'std_atomic_mass': 7.292250199264196,
'wtd_range_fie': 7.151624951194831,
'entropy_FusionHeat': 7.069847960411798,
'Te': 6.975228611552435,
'wtd_entropy_atomic_mass': 6.877664441886284,
'S': 6.574626605353182,
'gmean_Valence': 6.339719548522305,
'Xe': 5.596812559503841,
'Pb': 5.445375395590002,
'Au': 5.442173927520701,
'Tb': 5.3847712110203245,
'wtd_std_ThermalConductivity': 5.166339452102838,
'gmean_ElectronAffinity': 5.161011304703671,
'Hg': 5.025052094297533,
'mean_fie': 5.003860109194855,
'Ne': 4.930873010747075,
'wtd_std_atomic_mass': 4.71937808421278,
'wtd_range_FusionHeat': 4.500223329140536,
'mean_FusionHeat': 4.4143396479120405,
'mean_atomic_radius': 4.404995641865948,
'entropy_ThermalConductivity': 4.368220980912161,
'Ir': 4.249959602459216,
'Yb': 3.8953550721372396,
'mean_Valence': 3.7089432296434546,
'std_Valence': 3.4876652164013864,
'wtd_range_ElectronAffinity': 3.4707848762762006,
```

```
'gmean_ThermalConductivity': 3.3282625728437782,
'wtd_entropy_ElectronAffinity': 3.1717582264839423,
'Dy': 3.0534887247597147,
'gmean_fie': 3.038235132476612,
'std_Density': 3.008688992839782,
'N': 2.858886442856935,
'wtd_entropy_atomic_radius': 2.810801374766463,
'range_Density': 2.790487002440247,
'wtd_range_atomic_mass': 2.7691211665542204,
'range_FusionHeat': 2.7545665374504558,
'Tm': 2.7275421938545423,
'wtd_mean_Valence': 2.6356594054954847,
'wtd_range_atomic_radius': 2.6293706373414825,
'gmean_Density': 2.535894783973816,
'La': 2.3626927332742036,
'entropy_ElectronAffinity': 2.335213124448163,
'Ho': 2.2972922880342668,
'Pr': 2.251053708318047,
'mean_ElectronAffinity': 2.2137345508615316,
'He': 2.0398007287297544,
'wtd_entropy_Density': 1.9261539192964232,
'Tl': 1.8257114916233292,
'Mo': 1.8176182121205717,
'wtd_entropy_ThermalConductivity': 1.7887015013073493,
'Sm': 1.774728691803154,
'Ge': 1.7366025169716748,
'Gd': 1.7288076709844695,
'entropy_Density': 1.659807813091683,
'Al': 1.6352381758607275,
'Ni': 1.6205404380715394,
'P': 1.616029813679954,
'critical_temp': 1.5903667437812903,
'K': 1.525218526578877,
'wtd_range_Density': 1.4918656926198284,
'wtd_range_Valence': 1.3921665499578924,
'Si': 1.371165978901978,
'Re': 1.3439025629210517,
'Cd': 1.297312846219618,
'wtd_std_FusionHeat': 1.2295793803452253,
'Mn': 1.1298896361967758,
'mean_ThermalConductivity': 1.0676355781628066,
'gmean_atomic_mass': 1.0402777247214938,
'Ga': 1.0362997224882886,
'wtd_std_fie': 0.9813641226288073,
'As': 0.9610188881101238,
'number_of_elements': 0.9577835001175794,
'wtd_gmean_Density': 0.9483264485778177,
'Be': 0.8962827158736726,
'C': 0.8572348411535096,
'gmean_FusionHeat': 0.7572585387106064,
'wtd_std_Density': 0.7400832860166121,
'Cr': 0.6573503253831036,
'Eu': 0.6453994001876798,
'Sb': 0.6428258884186397,
'Lu': 0.556428177380533,
'Pt': 0.5183404169063515,
'Fe': 0.49014291191209514,
```

```
'Se': 0.4495194031822201,
'Zn': 0.41597531590190345,
'Co': 0.3909282510461157,
'wtd_gmean_Valence': 0.3748898887573451,
'std_FusionHeat': 0.374596539587452,
'V': 0.3402253112823565,
'Pm': 0.29973232715150905,
'Ca': 0.27787659202703113,
'Tc': 0.26986341348455223,
'Sr': 0.2401181674518157,
'Ta': 0.22227735447628277,
'Sn': 0.20931911858524127,
'Li': 0.2070881397680395,
'Na': 0.19230246477019114,
'Mg': 0.16677337486842167,
'Rb': 0.15981629299843023,
'wtd_gmean_fie': 0.14770698615424116,
'Nb': 0.11219837900968294,
'Os': 0.09359431086609563,
'W': 0.09266923248615051,
'Zr': 0.08971307499131446,
'Cu': 0.06776240155316159,
'Hf': 0.06568032258619723,
'Sc': 0.06333801909648593,
'B': 0.05375366268827595,
'Ru': 0.05062087289089685,
'Ce': 0.049267036373158074,
'In': 0.047553454427502306,
'Rh': 0.04718380802698852,
'Ba': 0.03680353296040104,
'Ti': 0.027092838316867224,
'Y': 0.0045853718116380905,
'H': 0.0,
'F': 0.0,
'Cl': 0.0,
'Br': 0.0,
'I': 0.0,
'Nd': 0.0,
'Er': 0.0,
'Bi': 0.0,
'Po': 0.0,
'At': 0.0}
```

Ridge Coefficients

In [ ]:
```
ridge_train.best_estimator_['ridge'].coef_
```

Out[ ]:
```
array([ 1.20324413e+00,  2.20346450e+01, -4.03065560e+01, -6.43441155e+00,
        2.42649729e+01, -1.23044395e+01,  2.44146902e+00,  1.41406662e+01,
        2.82145340e+00, -1.17358604e+01, -2.67478182e-01, -1.52923755e+01,
        4.57257803e+00,  1.36616003e+01,  4.82461363e+00, -2.14090017e+00,
        1.44174043e+01,  2.75633752e+01,  8.35245059e+00, -1.94135515e+01,
       -5.49363845e+00, -7.80156981e+00,  7.62310037e+01,  1.66167608e+00,
       -8.25042373e+01, -1.10509209e+01,  1.20629520e+01,  9.62290305e+00,
       -2.47657895e+00, -1.27932472e+01,  1.66552784e+00, -1.12203557e+01,
        1.69822669e+01,  2.17652782e+00, -2.20888804e+00, -1.53948065e+00,
       -2.18476427e+00, -2.68867123e+00, -1.48661728e+00,  3.40379669e+00,
       -1.15082410e+00, -1.77067822e+00,  1.53468318e+01,  4.85220553e+00,
       -1.90720584e+01,  2.83361737e+00, -3.38763584e+00, -1.78942371e+01,
       -3.27595533e+00,  2.09907883e+01, -1.04204233e+01,  1.12556724e+01,
       -2.32438071e+01, -7.31315106e+00,  1.85161509e+01, -1.04081962e+01,
        1.14808700e+01, -2.03651088e+00,  5.05646180e+00, -3.95770684e+00,
        2.77554515e+00, -1.80387481e+00,  2.05960475e+01, -2.70970471e+00,
       -1.41494817e+01,  5.85949490e+00,  8.48953347e-01, -3.19735952e+01,
       -1.12471641e+01,  3.35231118e+01,  3.06675980e+00,  9.49809581e+00,
       -1.39530319e+01, -4.32589850e+00,  9.30282386e+00,  3.07334771e+01,
       -3.60100904e+01,  1.28969008e+01, -2.14515582e+00, -3.13781443e+00,
       -1.15825733e+01, -5.92094514e-01,  0.00000000e+00,  2.72399882e+00,
       -3.70482965e-01, -1.02466061e+00, -6.55805352e-02, -3.35754197e-01,
       -3.19839342e+00,  9.15132226e+00,  0.00000000e+00,  6.87830668e+00,
        4.44699545e-01, -1.30841389e-01, -1.61286838e+00, -1.54644489e+00,
       -1.70765800e+00, -5.04900290e+00,  0.00000000e+00,  9.49594799e+00,
        1.64150126e+00,  8.47822755e-02, -3.66268712e-02,  3.96890356e-02,
       -3.23138798e-01, -6.04680068e-01,  1.13151166e+00, -4.86576010e-01,
       -4.12884952e-01, -1.56897943e+00, -1.09542386e-01,  3.92842832e-01,
       -1.05760881e+00, -1.73041629e+00, -9.26657745e-01,  1.22348715e+00,
        0.00000000e+00,  9.88329567e+00,  2.99842115e-01, -3.07663866e-01,
        8.76579365e-03,  8.76317644e-02,  1.27955887e-01,  1.99791138e+00,
        3.08501245e-01, -5.66833737e-02,  5.24874917e-02, -1.08189756e+01,
       -1.31058436e+01,  1.15653445e+00, -7.30836442e-02, -1.80117700e-01,
        7.62693657e-01,  7.50936751e+00,  0.00000000e+00,  7.14703854e+00,
        1.16600744e+01, -2.16017250e-02, -2.44367723e+00, -5.23680313e-02,
       -1.98780854e+00,  0.00000000e+00,  3.35297412e-02, -1.92495212e+00,
       -7.09207660e-01,  1.65958931e+00,  5.32793106e+00,  2.86817339e+00,
        2.19810018e+00,  1.80804633e-02,  2.19567468e+00,  3.88613319e+00,
       -5.20957163e-01, -8.61567810e-02,  6.49733832e-01, -9.20539523e-02,
        1.27131574e+00,  1.20267673e-01,  3.97914357e+00, -4.58660680e-01,
        5.78202763e+00,  5.09142213e+00,  1.78190081e+00,  5.49509654e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00])
```

In [ ]:
```
ridge_weights = {df.columns[key]:abs(value) for key, value in enumerate(ridg
dict(sorted(ridge_weights.items(), key=lambda item: item[1], reverse=True))
```

Out[ ]:
```
{'wtd_gmean_atomic_radius': 82.50423733510665,
 'wtd_mean_atomic_radius': 76.23100370069614,
 'wtd_mean_atomic_mass': 40.30655598944086,
 'wtd_entropy_Valence': 36.010090382187286,
 'std_ThermalConductivity': 33.52311183863332,
 'range_ThermalConductivity': 31.973595153364414,
 'entropy_Valence': 30.733477135522136,
 'range_fie': 27.563375166707345,
 'wtd_gmean_atomic_mass': 24.26497287703653,
 'wtd_mean_FusionHeat': 23.24380714242241,
 'mean_atomic_mass': 22.03464500407608,
```

```
'std_ElectronAffinity': 20.9907882840034,
'wtd_mean_ThermalConductivity': 20.596047543511407,
'std_fie': 19.413551483808554,
'wtd_gmean_ElectronAffinity': 19.07205841254293,
'wtd_gmean_FusionHeat': 18.516150931227177,
'range_ElectronAffinity': 17.894237113706332,
'wtd_mean_Density': 16.982266859423316,
'wtd_mean_ElectronAffinity': 15.346831797066539,
'mean_fie': 15.292375456462747,
'wtd_entropy_fie': 14.417404282907794,
'wtd_gmean_ThermalConductivity': 14.149481717128543,
'range_atomic_mass': 14.140666241169422,
'wtd_mean_Valence': 13.953031890072314,
'gmean_fie': 13.661600275605622,
'Ag': 13.105843630481445,
'range_Valence': 12.896900802682287,
'std_atomic_radius': 12.793247170520273,
'entropy_atomic_mass': 12.304439463852427,
'wtd_entropy_atomic_radius': 12.062951965890003,
'std_atomic_mass': 11.735860413275395,
'Cs': 11.6600744382917,
'wtd_std_Valence': 11.582573279653054,
'wtd_entropy_FusionHeat': 11.480869992011158,
'mean_FusionHeat': 11.255672400613115,
'wtd_range_ThermalConductivity': 11.247164099420255,
'mean_Density': 11.22035571751498,
'entropy_atomic_radius': 11.050920938098947,
'Pd': 10.818975577499792,
'wtd_std_ElectronAffinity': 10.420423262643894,
'entropy_FusionHeat': 10.408196159105955,
'Kr': 9.883295671758162,
'range_atomic_radius': 9.622903050655621,
'mean_Valence': 9.498095813038935,
'Ar': 9.495947985901113,
'wtd_gmean_Valence': 9.30282385733451,
'O': 9.151322262798242,
'wtd_range_fie': 8.352450586707699,
'mean_atomic_radius': 7.801569810205453,
'Te': 7.509367514891667,
'gmean_FusionHeat': 7.313151063543804,
'Xe': 7.147038543619432,
'Ne': 6.878306683019584,
'gmean_atomic_mass': 6.434411547223428,
'entropy_ThermalConductivity': 5.859494902934971,
'Au': 5.7820276277569915,
'Pb': 5.495096544551145,
'wtd_std_fie': 5.493638453572816,
'Tb': 5.3279310575015355,
'Hg': 5.091422127436061,
'wtd_range_FusionHeat': 5.056461796068855,
'S': 5.0490029024770955,
'gmean_ElectronAffinity': 4.852205532799202,
'wtd_gmean_fie': 4.824613626594694,
'wtd_mean_fie': 4.572578033628301,
'gmean_Valence': 4.325898500560551,
'Ir': 3.9791435693446893,
'std_FusionHeat': 3.957706838806994,
```

```
'Yb': 3.886133185279942,
'std_Density': 3.403796691747466,
'wtd_entropy_ElectronAffinity': 3.3876358417785863,
'wtd_range_ElectronAffinity': 3.2759553278455487,
'N': 3.1983934169938997,
'std_Valence': 3.137814432358151,
'wtd_std_ThermalConductivity': 3.066759795578775,
'Dy': 2.8681733896871577,
'entropy_ElectronAffinity': 2.83361737034622,
'wtd_range_atomic_mass': 2.8214534021231765,
'wtd_std_FusionHeat': 2.775545145152581,
'He': 2.7239988209679544,
'gmean_ThermalConductivity': 2.7097047060217156,
'range_Density': 2.688671229516349,
'wtd_range_atomic_radius': 2.476578947198765,
'La': 2.44367722887337,
'wtd_entropy_atomic_mass': 2.441469015768297,
'wtd_gmean_Density': 2.2088880386686385,
'Ho': 2.198100184514838,
'Tm': 2.1956746756273757,
'wtd_entropy_Density': 2.18476426849969,
'gmean_Density': 2.176527815541163,
'wtd_range_Valence': 2.1451558154296495,
'entropy_fie': 2.1409001732245008,
'range_FusionHeat': 2.0365108825989995,
'Mo': 1.9979113762295582,
'Pr': 1.9878085390622042,
'Sm': 1.9249521213067178,
'mean_ThermalConductivity': 1.8038748071839,
'Tl': 1.7819008083846928,
'mean_ElectronAffinity': 1.770678224956746,
'Ge': 1.7304162929630587,
'P': 1.7076580005004118,
'wtd_std_atomic_radius': 1.665527837265629,
'gmean_atomic_radius': 1.661676080697901,
'Gd': 1.6595893067067595,
'K': 1.641501256750352,
'Al': 1.6128683757915894,
'Ni': 1.5689794280307592,
'Si': 1.5464448895433245,
'entropy_Density': 1.5394806548454576,
'wtd_range_Density': 1.4866172764681747,
'Re': 1.2713157378617894,
'Se': 1.2234871458145533,
'number_of_elements': 1.2032441290584106,
'Cd': 1.1565344516443263,
'wtd_std_Density': 1.1508241026344765,
'Mn': 1.131511662090882,
'Ga': 1.0576088138978954,
'Be': 1.0246606126999513,
'As': 0.9266577450597306,
'wtd_entropy_ThermalConductivity': 0.8489533472971035,
'Sb': 0.7626936574273563,
'Eu': 0.7092076601274739,
'Ta': 0.6497338324638403,
'Cr': 0.6046800680790798,
'critical_temp': 0.5920945136520607,
```

'Lu': 0.5209571626836677,
'Fe': 0.4865760097630898,
'Pt': 0.4586606798933014,
'Na': 0.4446995452341047,
'Co': 0.41288495225960375,
'Zn': 0.39284283160655326,
'Li': 0.37048296493134236,
'C': 0.33575419739723905,
'V': 0.3231387979596321,
'Tc': 0.3085012451591725,
'Sr': 0.3076638659470038,
'Rb': 0.29984211474783373,
'wtd_std_atomic_mass': 0.26747818172496235,
'Sn': 0.18011769958809024,
'Mg': 0.1308413892168927,
'Nb': 0.1279558868584092,
'Os': 0.12026767318856164,
'Cu': 0.10954238558510367,
'W': 0.09205395231182723,
'Zr': 0.08763176437368028,
'Hf': 0.08615678102718098,
'Ca': 0.08478227548101774,
'In': 0.07308364419531818,
'B': 0.06558053522385868,
'Ru': 0.056683373705555276,
'Rh': 0.05248749168577969,
'Ce': 0.05236803128972188,
'Ti': 0.039689035605733836,
'Sc': 0.03662687117883952,
'Pm': 0.033529741194744334,
'Ba': 0.02160172499564086,
'Er': 0.018080463299312347,
'Y': 0.00876579364589835,
'H': 0.0,
'F': 0.0,
'Cl': 0.0,
'Br': 0.0,
'I': 0.0,
'Nd': 0.0,
'Bi': 0.0,
'Po': 0.0,
'At': 0.0}