**Case Study Report: Superconductors Analysis**

**Problem Statement**

The objective of this case study is to analyze a dataset containing physical and chemical properties of materials and predict their critical temperature (‘critical\_temp’) for superconductivity. The analysis involves preprocessing the data, building a predictive model, and interpreting the results to gain insights into the factors influencing superconductivity.

The dataset contains 81 features and a target variable (‘critical\_temp’). These features represent various physical and chemical attributes such as atomic mass, electron affinity, and thermal conductivity, among others.

**Preprocessing Steps**

1. **Normalization**:
   * All features were normalized using StandardScaler to ensure consistent scales across the dataset. This step is crucial for models sensitive to feature scaling.
2. **Handling Missing Values**:
   * The dataset contained no missing values, as verified by data.isnull().sum().
3. **Train-Test Split**:
   * The dataset was split into training (80%) and testing (20%) sets to evaluate the model’s performance on unseen data.

**Model Development**

1. **Model Selection**:
   * Linear Regression was chosen as the initial model to predict the critical temperature. This model provides a straightforward way to interpret feature importance through coefficients.
2. **Training**:
   * The training set was used to fit the model, and the coefficients and intercept were determined.
3. **Evaluation**:
   * The testing set was used to evaluate the model’s performance. Metrics such as Mean Squared Error (MSE) and R² were calculated.

**Results**

1. **Evaluation Metrics**:
   * Mean Squared Error (MSE): 0.257
   * R² Score: 0.738
2. **Visualizations**:
   * **Residuals Plot**: The residuals are distributed around zero, suggesting a reasonably good fit, though some heteroscedasticity is evident.
   * **Predicted vs. Actual Plot**: The predicted values align closely with the actual values, further confirming the model’s performance.
3. **Feature Importance**:
   * Top Positive Contributors:
     + wtd\_mean\_atomic\_radius (2.43)
     + entropy\_Valence (0.84)
     + std\_ElectronAffinity (0.82)
   * Top Negative Contributors:
     + wtd\_gmean\_atomic\_radius (-2.60)
     + entropy\_fie (-1.12)
     + wtd\_mean\_FusionHeat (-0.85)

**Discussion and Recommendations**

1. **Scientific Insights**:
   * The weighted mean atomic radius is the most significant positive contributor to critical temperature. This aligns with scientific findings that atomic structure plays a crucial role in superconductivity.
   * Entropy-related features indicate that disorder within the material’s properties significantly influences superconducting behavior.
2. **Recommendations**:
   * Further studies could focus on optimizing material compositions based on key features like atomic radius and electron affinity.
   * Non-linear models such as Random Forests or Gradient Boosting can be explored for better performance.

**Conclusion**

The analysis demonstrated the utility of a Linear Regression model in predicting critical temperatures for superconductors. While the model performed well with an R² of 0.738, further refinement using advanced models could enhance prediction accuracy. The feature importance analysis provided meaningful scientific insights, paving the way for future research into material properties and superconductivity.

**Code Appendix**

# Import necessary libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

data = pd.read\_csv('superconductors\_data.csv')

# Display basic dataset information

print(data.info())

print(data.describe())

# Check for missing values

missing\_values = data.isnull().sum()

print("Missing Values:\n", missing\_values)

# Normalize the dataset

scaler = StandardScaler()

normalized\_data = pd.DataFrame(scaler.fit\_transform(data), columns=data.columns)

# Define features and target

target\_column = 'critical\_temp'

X = normalized\_data.drop(columns=[target\_column])

y = normalized\_data[target\_column]

# Split into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize the linear regression model

model = LinearRegression()

# Train the model

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate performance

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R^2 Score: {r2}")

# Extract coefficients and intercept

coefficients = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef\_})

coefficients.sort\_values(by='Coefficient', ascending=False, inplace=True)

print("Feature Importances:\n", coefficients)

print(f"Intercept: {model.intercept\_}")

# Plot residuals

residuals = y\_test - y\_pred

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, residuals, alpha=0.6)

plt.axhline(0, color='red', linestyle='--')

plt.title("Residuals Plot")

plt.xlabel("Actual Values")

plt.ylabel("Residuals")

plt.show()

# Predicted vs. Actual scatterplot

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, alpha=0.6, color='blue')

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], color='red', linestyle='--')

plt.title("Predicted vs. Actual")

plt.xlabel("Actual Values")

plt.ylabel("Predicted Values")

plt.show()