Geometallurgy Introduction: Kriging Method and Machine Learning.

Geometallurgy is an interdisciplinary field that combines geological, mineralogy, and metallurgical information to improve the predictability, efficiency, and sustainability of mineral processing and mining operations. The goal of geometallurgical studies is to build a model that can accurately predict the behavior of the ore during the mining and processing stages and to provide guidance for future exploration, mine planning, and risk management. This predictive model can help mining companies optimize their operations, reduce environmental impact, and increase profits.

A typical geometallurgical model integrates various data types, including geological data (for example, the size of the ore body, shape, and grade distribution), mineralogical data (for example, the types and distribution of minerals), and metallurgical data (i.e., how the ore responds to different metallurgical and processing methods). The modeling process generally involves data collection, statistical analysis, model building, and validation. The outputs of a geometallurgical model can include predictions of ore grade, throughput, recovery, concentration quality, and other relevant metallurgical parameters.

One standard method used in building geometallurgical models is multivariate statistical analysis, specifically, Principal Component Analysis (PCA) and Multiple Linear Regression (MLR). For instance, an MLR model can predict a metallurgical response variable (Y) based on several predictor variables (X1, X2, ..., Xn). The general form of MLR is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ \ldots\ldots\ldots \beta_n X_n + \varepsilon$$

Where β_0 is the intercept, β_1, β_2, ..., β_n are the coefficients, and ε is the error term. The coefficients are estimated by minimizing the sum of the squared residuals.

Geostatistical simulation techniques often use a standard method called conditional simulation. This generates multiple models of the spatial distribution of a variable, all with equal probability. The resulting distribution of possible outcomes can be useful for risk analysis. One example of a conditional simulation equation is the Kriging method, which is shown below:

$$Z^*(x_0) = \sum_i \lambda_i Z(x_i)$$

Here $Z^*(x_0)$ is the estimated grade at location x_0, and λ_i is the weight assigned to the grade $Z(x_i)$ at location x_i.

Geometallurgical modeling also employs machine learning algorithms to capture complex relationships between variables. Decision tree-based methods like Random Forest or Gradient Boosting are commonly used because they can handle large datasets, capture nonlinear relationships, and handle interactions between predictors. However, the accuracy of a geometallurgical model largely depends on the quality and quantity of input data. Comprehensive data collection is often challenging in practice due to the high drilling, sampling, and laboratory testing costs. Therefore, practical strategies for data collection, obtaining representative samples, and combining different types of data, are crucial.

An effective geometallurgical program also requires careful management of uncertainties, which can come from various sources, such as sampling errors, measurement errors, or geological uncertainties. These uncertainties must be quantified and propagated through the model using statistical and geostatistical methods.

Despite the advances that have been made in the field of geometallurgy, there is still room for further research.

In other words, geometallurgy is essential for sustainable and efficient metallurgical/mineral processing and mining. It involves integrating various data types and applying statistical, geostatistical, and machine learning methods to build predictive models of ore behavior. Just like any other field, the field offers ample opportunities for further research and improvement.

Kriging Technique

Kriging is a geostatistical interpolation technique that considers both the distance and the degree of variation between known data points when estimating values in unknown areas. It assigns weights to the surrounding measured values such that the variance of the prediction error is minimized.

Below is illustration of a simplified example of Kriging model in Python.

```
from pykrige.ok import OrdinaryKriging
import numpy as np

# The locations of your samples (longitudes, latitudes)
lon = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
lat = np.array([1.0, 2.0, 3.0, 4.0, 5.0])

# The grades at your sample locations
grade = np.array([10.0, 15.0, 20.0, 25.0, 30.0])

# Create the Ordinary Kriging object.
# The parameters are the coordinates (x=longitude, y=latitude), z=values,
# variogram_model specifies the variogram model to use ('spherical', 'exponential', etc),
# and nlags is the number of lags to consider in the semivariogram.
OK = OrdinaryKriging(lon, lat, grade, variogram_model='linear', nlags=6)

# The locations where you want to estimate grades
grid_lon = np.array([1.5, 2.5, 3.5, 4.5])
grid_lat = np.array([1.5, 2.5, 3.5, 4.5])

# Execute on the grid
z, ss = OK.execute('grid', grid_lon, grid_lat)

# z are the estimated grades at the grid points
```

```
# ss are the estimated variances of the predictions at the grid points
print("Estimated Grades:", z)
print("Estimated Variances:", ss)
```

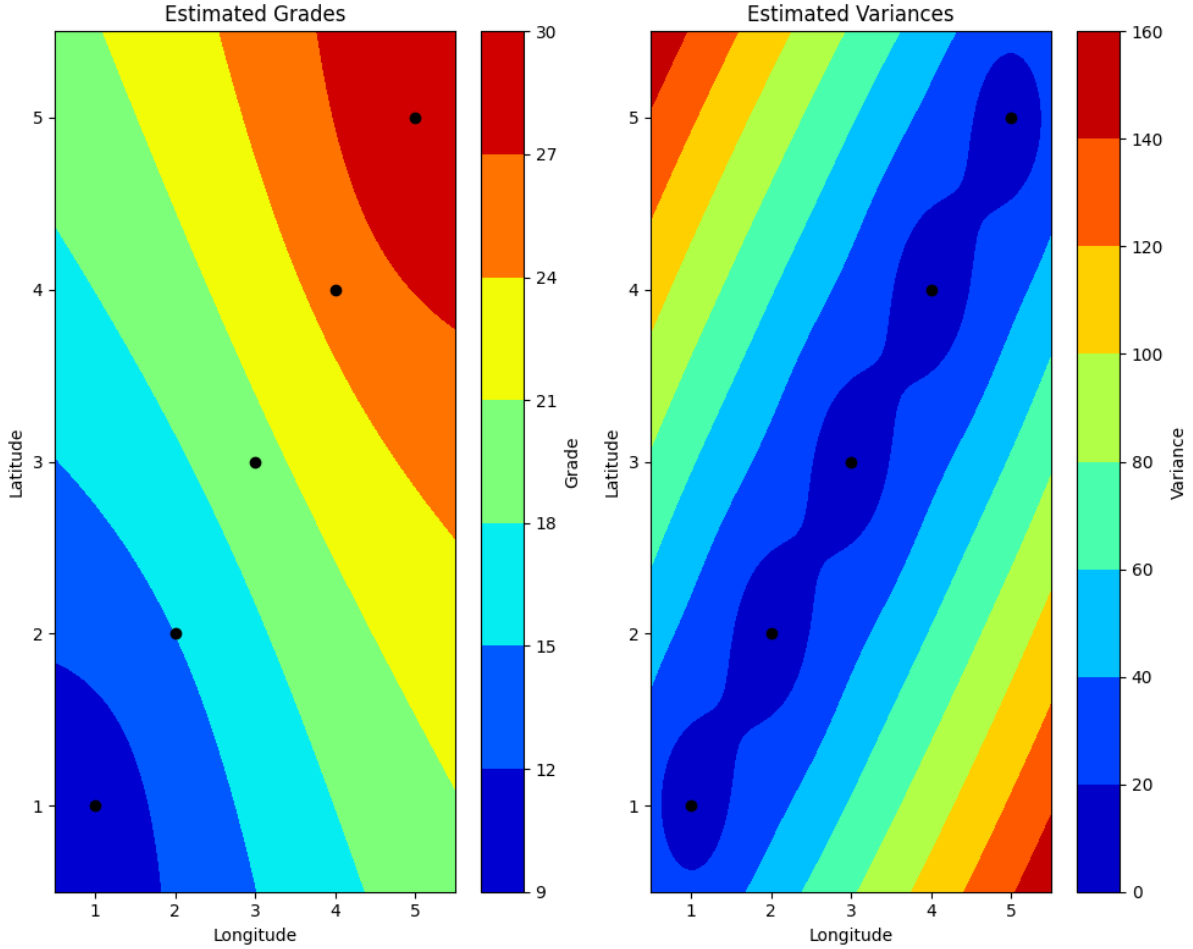
```
Estimated Grades: [[12.499999999999995 15.191631809001953 17.775483085411853 20.0]
 [15.191631809001953 17.499999999999993 20.000000000000004
  22.224516914588143]
 [17.775483085411853 20.000000000000004 22.500000000000007
  24.80836819099804]
 [20.0 22.224516914588143 24.80836819099804 27.500000000000004]]
Estimated Variances: [[18.809027975964376 32.06199920734323 61.340335902436806
  91.11864704866474]
 [32.06199920734323 18.809027975964383 32.00964932065947
  61.340335902436834]
 [61.340335902436806 32.00964932065947 18.809027975964433
  32.06199920734331]
 [91.11864704866474 61.340335902436834 32.06199920734331
  18.809027975964426]]
```

Plotting

```
from pykrige.ok import OrdinaryKriging
import numpy as np
import matplotlib.pyplot as plt

# The locations of your samples (longitudes, latitudes)
lon = np.array([1.0, 2.0, 3.0, 4.0, 5.0])
lat = np.array([1.0, 2.0, 3.0, 4.0, 5.0])

# The grades at your sample locations
grade = np.array([10.0, 15.0, 20.0, 25.0, 30.0])

# Creating the Ordinary Kriging object.
OK = OrdinaryKriging(lon, lat, grade, variogram_model='linear', nlags=6)

# Define a grid of points where you want to estimate grades.
grid_lon = np.linspace(0.5, 5.5, 100)
grid_lat = np.linspace(0.5, 5.5, 100)

# Execute on the grid
z, ss = OK.execute('grid', grid_lon, grid_lat)

# Plot for estimated grades
plt.figure(figsize=(10,8))
plt.subplot(121)
plt.title("Estimated Grades")
```

```
plt.contourf(grid_lon, grid_lat, z, cmap='jet')
plt.colorbar(label='Grade')
plt.scatter(lon, lat, color='black')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# Plot for estimated variances
plt.subplot(122)
plt.title("Estimated Variances")
plt.contourf(grid_lon, grid_lat, ss, cmap='jet')
plt.colorbar(label='Variance')
plt.scatter(lon, lat, color='black')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

plt.tight_layout()
plt.show()
```

[2]



Machine learning algorithms in improving accuracy and efficiency of Geometallurgical Modeling.

Machine learning (ML) algorithms can be used in geometallurgical modeling to discover complex, non-linear relationships between different variables that are difficult to identify with traditional statistical methods. For example, ML algorithms can be used to model the relationship between geological and mineralogical features and an ore's metallurgical behavior, predict the yield from different processing methods, or estimate the grade of ore based on exploration data. Several types of ML algorithms can be applied to geometallurgical data, including decision trees, random forests, support vector machines, and neural networks. The choice of algorithm depends on the nature of the problem, the type and amount of data available, and the application's specific requirements.

The application of ML in geometallurgical modeling involves several steps:

1. **Data Preparation**: This includes data cleaning, handling missing data, feature engineering, and feature scaling. Data quality is often a crucial factor for the performance of ML algorithms.
2. **Model Training**: In this step, the ML algorithm is trained on a subset of the data (the training set). The goal is to learn a function to predict the targeted response variable based on the input variables.
3. **Model Validation**: The trained model is tested on a different subset of the data (the validation set) to evaluate its performance and tune the hyperparameters. This helps ensure the model is not overfitting or underfitting the training data.
4. **Model Testing**: The final model is evaluated on a third subset of the data (the test set) that was not used during the training and validation. It provides an unbiased estimate of the model's performance.
5. **Model Deployment**: Once the model has been tested and validated, it can be used to predict new data.

Here are some best practices for training, testing, and validating ML models in geometallurgical modeling:

- **Data Splitting**: Splitting your data into separate training, validation, and test sets is essential. A common rule of thumb is to use 70% of the data for training, 15% for validation, and 15% for testing. The exact proportions can vary depending on the amount of data you have.
- **Cross-Validation**: Cross-validation is a technique for more robustly assessing the performance of ML models. It involves dividing the data into 'k' subsets, training the model on 'k-1' subsets, and testing it on the remaining subset. This process is repeated 'k' times so that each subset is used for testing once. The average performance across the 'k' iterations is taken as the model's performance.
- **Handling Imbalanced Data**: If the target variable has imbalanced classes (i.e., one class has many more examples than another), it can bias the ML model toward predicting the majority class. Techniques such as oversampling the minority class, undersampling the majority class, or using a combination of both (SMOTE) can be used to address this issue.
- **Feature Importance**: ML algorithms like random forest and gradient boosting can provide feature importance measures, which can be used to identify the most relevant variables for prediction. It is helpful for feature selection and gaining insights into the relationship between variables.

- **Regularization**: Regularization techniques, such as L1 and L2 regularization, can prevent overfitting by adding a penalty term to the loss function that the model minimizes.
- **Model Interpretability**: While complex ML models like neural networks can provide high accuracy, they are often described as "black boxes" because their predictions are difficult to interpret. If interpretability is important, simpler models like decision trees or methods for interpreting complex models like LIME or SHAP can be used.

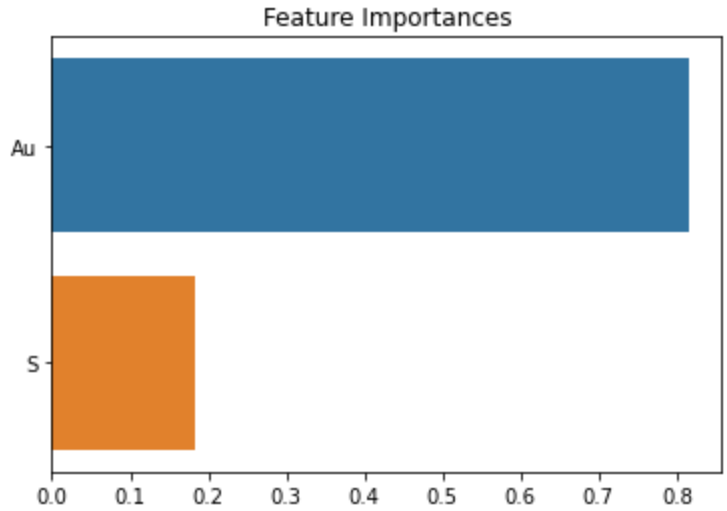
The application of ML in geometallurgy is an active area of research, and there are many opportunities for developing and applying new methods and techniques. As with any procedure, evaluating ML models' results and understanding their limitations is essential.

Machine learning can significantly enhance the accuracy and efficiency of geometallurgical modeling. A good example is classification or regression algorithms that can be used to predict the metallurgical response of an ore deposit based on geochemical and mineralogical data. Consider the case of predicting the metallurgical recovery of an ore deposit based on some geochemical features. We can use a random forest, a robust machine-learning algorithm that often works well on such projects, via the popular scikit-learn library. Let’s look at an example, note that the data used here is synthetic.

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns

# Creating a synthetic dataset
np.random.seed(123)
n_samples = 500

# Assume that we have two geochemical features
Au = np.random.normal(70, 20, n_samples)
S = np.random.normal(6, 1, n_samples)

# And our target variable is metallurgical recovery
Recovery = Au * 0.5 + S * 5 + np.random.normal(0, 2, n_samples)

df = pd.DataFrame({
    'Au': Au,
    'S': S,
    'Recovery': Recovery
})

# Split data into features (X) and target (y)
X = df[['Au', 'S']]
y = df['Recovery']

# Split the data 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 a random forest regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
rf.fit(X_train, y_train)

# Predict on the test data
y_pred = rf.predict(X_test)

# Compute the root mean squared error of our predictions
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE: ", rmse)

# Plot the feature importances
importances = rf.feature_importances_
sns.barplot(x=importances, y=X.columns)
plt.title('Feature Importances')
plt.show()
```

```
RMSE:  2.2631168578220717
```

```
RMSE:  2.2631168578220717
```



Geometallurgical Flotation Modeling Example

Geometallurgical modeling is a powerful tool that combines geological, mineralogical, and metallurgical data to predict how minerals will behave during processing accurately. It plays a crucial role in mine planning and mineral processing, promoting economically and environmentally sustainable mining operations. One of the critical principles of geometallurgical modeling is that ore characteristics - such as mineralogy, grain size, and associations - significantly impact how it behaves during processing. Consequently, processing methods for fine-grained gold ore may differ from those used for coarse gold ore. Geometallurgical models rely on a fundamental equation that predicts a specific mineral's metallurgical recovery (R) and is an invaluable tool for mining professionals:

$$R = f(X_1, X_2, \ldots\ldots, X_n)$$

Here, X_1, X_2, ..., X_n are different parameters representing the geological and mineralogical characteristics of the ore, and f is a function that predicts the metallurgical recovery based on these parameters. The specific form of the function f, as well as the parameters used, can vary depending on the type of ore, the mineral of interest, and the processing method used.

In practice, the function f is often determined using statistical or machine learning methods. For instance, a linear regression model might be used if the relationship between the parameters and the recovery is linear. A non-linear model, such as a neural network or a decision tree, might be used if the relationship is more complex.

Let's consider the case of a flotation plant. Flotation is a complex process with many variables affecting the recovery of sulfide minerals. These variables include mineralogical characteristics, chemical reagents, and operational conditions such as pH, temperature, pulp density, and air flow rate. For the sake of simplicity, let's consider a simplified example where the recovery (R) of a sulfide mineral in a flotation process is affected by the grade of the mineral (G), the pH of the

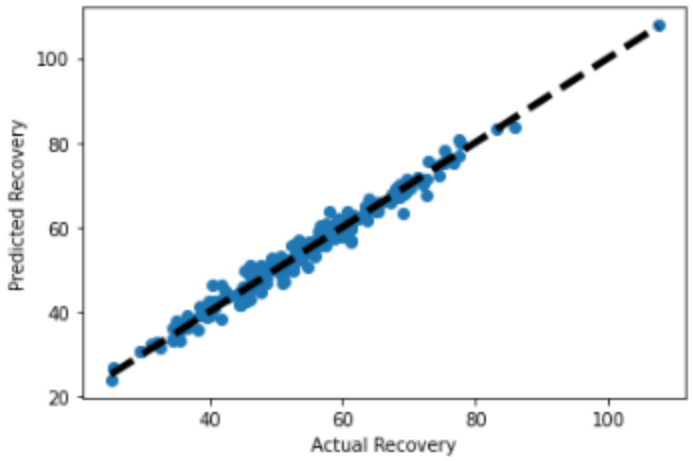
pulp (pH), and the concentration of the collector (C). A potential model for the recovery could be:

$$Recovery(R) = a * G + b * pH + c * C + d$$

This equation indicates that there is a straight connection between the recovery and the variables involved. The coefficients (a, b, and c) demonstrate how sensitive the recovery is to changes in the corresponding variables, while d is a constant. These coefficients are usually determined through experimental or operational data. The Python example below is a simplified illustration, but in reality, actual flotation models can be more intricate, taking into account nonlinear effects and interactions between variables that can be identified through DOE experiments.

In [1]:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(42)
n_samples = 1000

# Geological and mineralogical parameters
X1 = np.random.normal(80, 25, n_samples)  # e.g., Ni content
X2 = np.random.normal(3, 0.5, n_samples)  # e.g., S content

# Metallurgical recovery
R = X1 * 0.5 + X2 * 5 + np.random.normal(0, 2, n_samples)

df = pd.DataFrame({
    'X1': X1,
    'X2': X2,
    'R': R
})

# Split data into features (X) and target (y)
X = df[['X1', 'X2']]
y = df['R']

# Split the data 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 a linear regression model
lr = LinearRegression()

# Train the model
lr.fit(X_train, y_train)

# Predict on the test data
y_pred = lr.predict(X_test)

# Plot actual vs. predicted recovery
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Recovery')
plt.ylabel('Predicted Recovery')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
plt.show()
```

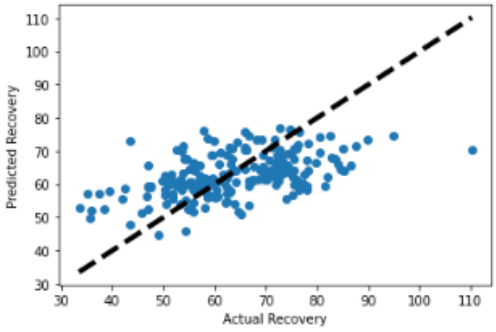


In [1]:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(42)
n_samples = 1000

# Mineral grade, pulp pH, and collector concentration
G = np.random.normal(5, 1, n_samples)
pH = np.random.uniform(7, 10, n_samples)
C = np.random.uniform(50, 200, n_samples)

# Metallurgical recovery
R = G * 5 + pH * 3 + C * 0.1 + np.random.normal(0, 10, n_samples)

df = pd.DataFrame({
    'G': G,
    'pH': pH,
    'C': C,
    'R': R
})

# Split data into features (X) and target (y)
X = df[['G', 'pH', 'C']]
y = df['R']

# Split the data 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 a linear regression model
lr = LinearRegression()

# Train the model
lr.fit(X_train, y_train)

# Print the coefficients
print("Grade Coefficient (a): ", lr.coef_[0])
print("pH Coefficient (b): ", lr.coef_[1])
print("Collector Concentration Coefficient (c): ", lr.coef_[2])
print("Constant (d): ", lr.intercept_)

# Predict on the test data
y_pred = lr.predict(X_test)

# Plot actual vs. predicted recovery
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Recovery')
plt.ylabel('Predicted Recovery')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
plt.show()
```

```
Grade Coefficient (a):  4.988365489043359
pH Coefficient (b):  2.7077224902291803
Collector Concentration Coefficient (c):  0.08658951172082613
Constant (d):  4.090870607430162
```

```
Grade Coefficient (a):  4.988365489043359
pH Coefficient (b):  2.7077224902291803
Collector Concentration Coefficient (c):  0.08658951172082613
Constant (d):  4.090870607430162
```



In this simplified model illustration, we create synthetic data for the grade (G), pulp pH, and collector concentration (C), and the recovery (R) based on our simplified model. We split this data into a training set and a testing set, then train a linear regression model on the training data. We print out the coefficients of the model, which represent the sensitivity of the recovery to the grade, pH, and collector concentration. Finally, we use the model to predict the recovery on the test data and plot the actual vs. the predicted recovery.

References

https://www.researchgate.net/publication/272002018_Providing_a_subsurface_reservoir_quality_maps_in_oil_fields_by_geostatistical_methods

Daniel Erdelyi , Istvan Gabor Hatvani , Hyeongseon Jeon, Matthew Jones , Jonathan Tyler, Zoltan Kern, “Predicting spatial distribution of stable isotopes in precipitation by classical geostatistical- and machine learning methods” Journal of Hydrology, Volume 617, Part C, February 2023, 129129

MATPLOTLIB: A 2D Graphics Environment By John D. Hunter<https://ieeexplore.ieee.org/document/4160265>

https://matplotlib.org/stable/users/index.html

M.A. Oliver, R. Webster “A tutorial guide to geostatistics: Computing and modelling variograms and kriging” CATENA Volume 113, February 2014, Pages 56-69

https://scholar.google.com/citations?user=EWO1fpoAAAAJ&hl=en

http://www.diva-portal.org/smash/get/diva2:1300831/FULLTEXT01.pdf