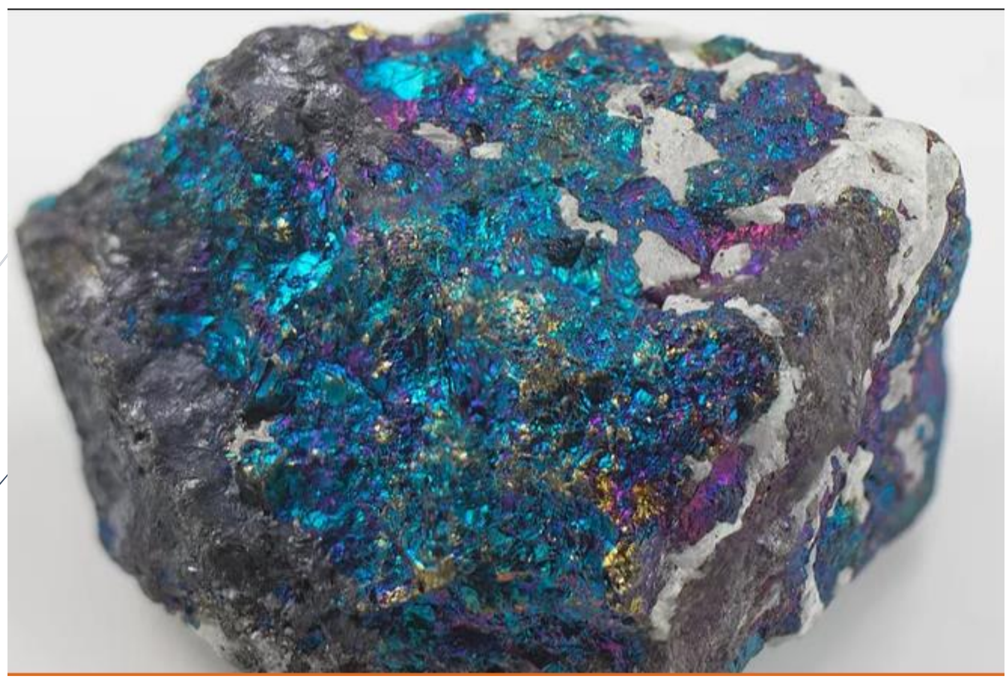
PREDICTIVE MODEL USING MACHINE LEARNING ALGORITHMS



Develop a predictive model using machine learning algorithms (supervised and unsupervised learning) to identify prospective targets for mining exploration.

Problem Statement

The process of Exploratory drilling and probing activities requires high investment, sustained cash inflow, and considerable time with inherent high risk.

Traditional procedures are unable to provide the degree of reliability/confidence limits. The application of mathematical models through data analysis can help improve precision and bring forth decision-making criteria at the end of each stage of the exploratory process.

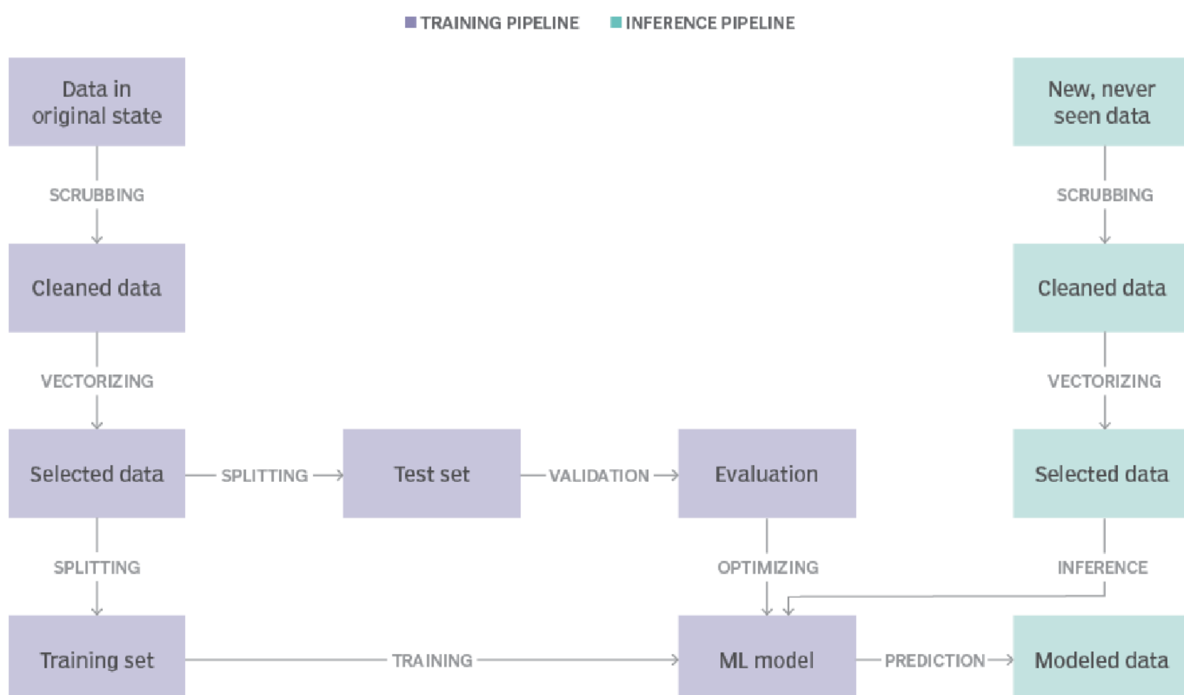
Objectives

To determine the economic viability of probable minerals deposit without the need for drilling and exploratory activities using machine learning algorithms.

Methodology

Based on the desired product output and considering our model performance metric and evaluation, KPIs and a final determination of whether the model can meet the established business goals we

Process



Data Cleaning

Data cleaning is the process of identifying incomplete, incorrect, inaccurate and irrelevant parts of the data and then replacing, modifying or deleting this dirty data. The data received contained about 11 CSV files and a file index explaining the meaning of the names of each file. The total number of **36 predictive variables** needed for the analysis and **1 dependent variable** Each feature is identified in each of the files. This includes Geochemistry (AU, CU, AS, PB, SB, ZN, NA, MG, MO, AG, FE, K, CA), Structural (Azimuth classification (3),w/o structure), Geophysics (Gravimetry & Magnetometry Rpt) and Lithology features (Type of rock (17))

#Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import normalize
```

Data Collection

Mineral forecast provided the dataset used for this algorithm

#Importing dataset into Jupyter environment

```
df = pd.read_csv('mineral_forecast.csv')
```

```
df.head()
```

```
Out[1]:

   Ag_ppm_AA_ICP  Au_ppb  Cu_ppm_AA_ICP  ...  SAMPTO  Soporte       s
0            1.6     NaN           56.0  ...   214.5      1.5  0.0010
1            0.6     NaN            8.0  ...   216.0      1.5  0.0010
2            0.4     NaN            8.0  ...   217.5      1.5  0.0010
3            0.4     NaN            5.0  ...   219.0      1.5  0.0010
4            0.6     NaN           17.0  ...   220.5      1.5  0.0009

[5 rows x 70 columns]
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178408 entries, 0 to 178407
Data columns (total 70 columns):
 #   Column           Non-Null Count   Dtype
```

```
---  ------            --------------   -----
 0   Ag_ppm_AA_ICP     1489 non-null    float64
 1   Au_ppb            834 non-null     float64
 2   Cu_ppm_AA_ICP     2639 non-null    float64
 3   As_ppm_AA_ICP     1854 non-null    float64
 4   Pb_ppm_AA_ICP     2372 non-null    float64
 5   Sb_ppm_AA_ICP     873 non-null     float64
 6   Zn_ppm_AA_ICP     2609 non-null    float64
 7   Na_pct_ICP        1150 non-null    float64
 8   Mg_pct_ICP        1167 non-null    float64
 9   Mo_ppm_AA_ICP     2135 non-null    float64
 10  Ag_ppm_AA_ICP.1   1489 non-null    float64
 11  Fe_pct_ICP        1185 non-null    float64
 12  K_pct_ICP         1185 non-null    float64
 13  Ca_pct_ICP        1177 non-null    float64
 14  X                 25425 non-null   float64
 15  Y                 25425 non-null   float64
 16  Clas_Rumb         25425 non-null   float64
 17  Type              25425 non-null   object
 18  Gravemetry        11609 non-null   float64
 19  Magnetometry RTP  46409 non-null   object
 20  0X                546 non-null     float64
 21  0Y                546 non-null     float64
 22  0Z                546 non-null     float64
 23  3X                34685 non-null   float64
 24  3Y                34685 non-null   float64
 25  3Z                34685 non-null   float64
 26  4X                5685 non-null    float64
 27  4Y                5685 non-null    float64
 28  4Z                5685 non-null    float64
```

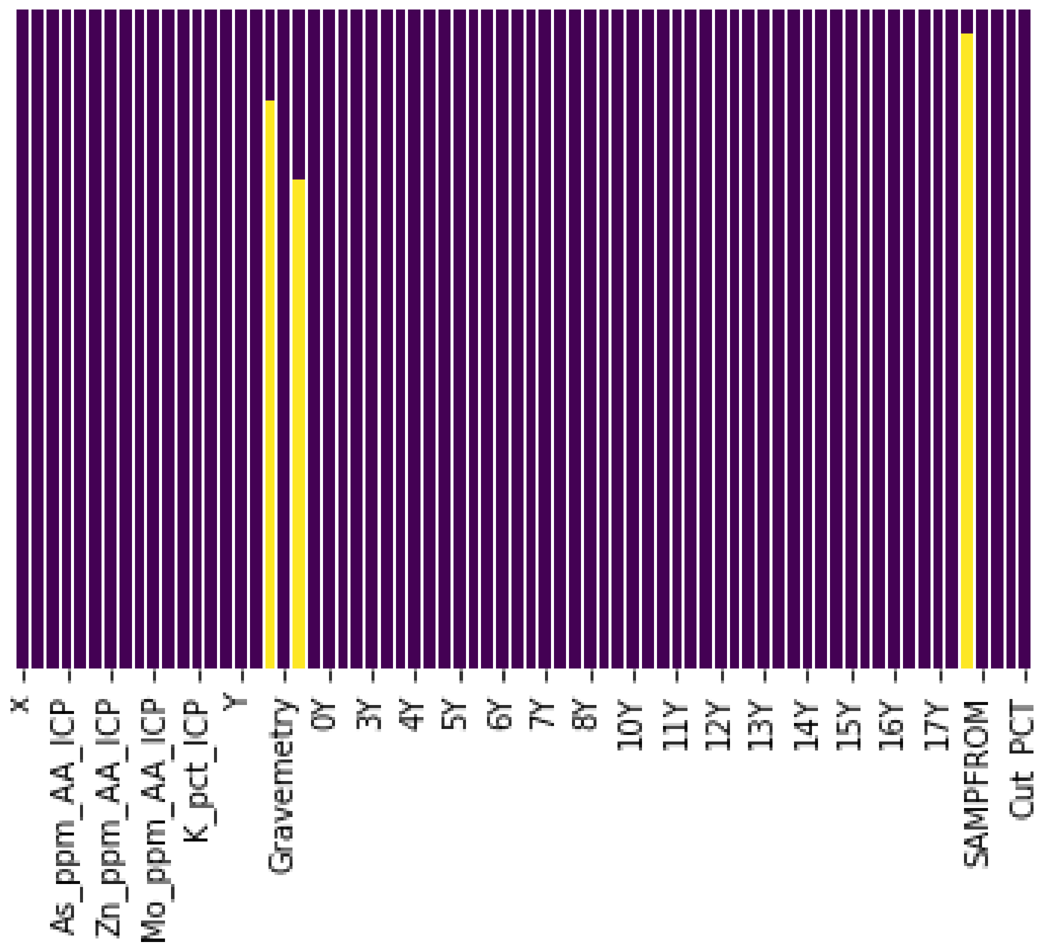
29	5X	9226	non-null	float64
30	5Y	9226	non-null	float64
31	5Z	9226	non-null	float64
32	6X	178408	non-null	int64
33	6Y	178408	non-null	int64
34	6Z	178408	non-null	float64
35	7X	130	non-null	float64
36	7Y	130	non-null	float64
37	7Z	130	non-null	float64
38	8X	48666	non-null	float64
39	8Y	48666	non-null	float64
40	8Z	48666	non-null	float64
41	10X	13588	non-null	float64
42	10Y	13588	non-null	float64
43	10Z	13588	non-null	float64
44	11X	11307	non-null	float64
45	11Y	11307	non-null	float64
46	11Z	11307	non-null	float64
47	12X	138733	non-null	float64
48	12Y	138733	non-null	float64
49	12Z	138733	non-null	float64
50	13X	106374	non-null	float64
51	13Y	106374	non-null	float64
52	13Z	106374	non-null	float64
53	14X	931	non-null	float64
54	14Y	931	non-null	float64
55	14Z	931	non-null	float64
56	15X	874	non-null	float64
57	15Y	874	non-null	float64
58	15Z	874	non-null	float64

```
 59  16X            16358 non-null   float64
 60  16Y            16358 non-null   float64
 61  16Z            16358 non-null   float64
 62  17X            158 non-null     float64
 63  17Y            158 non-null     float64
 64  17Z            158 non-null     float64
 65  HOLE ID        7405 non-null    object
 66  SAMPFROM       7405 non-null    float64
 67  SAMPTO         7405 non-null    float64
 68  Soporte        7405 non-null    float64
 69  Cut_Pct        7337 non-null    float64
```

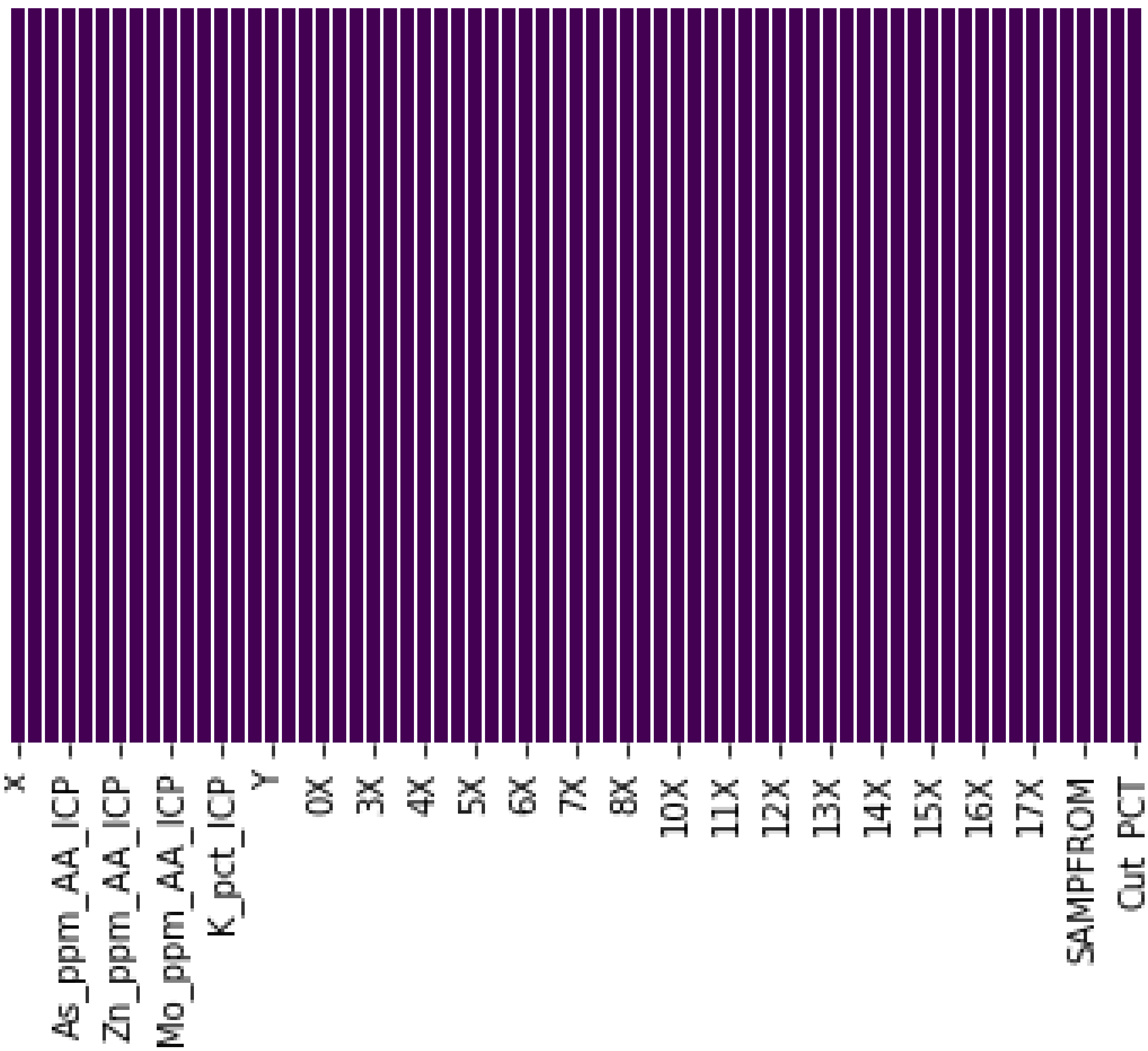
```
df.isnull().sum()
Out[17]:
Ag_ppm_AA_ICP    176919
Au_ppb           177574
Cu_ppm_AA_ICP    175769
As_ppm_AA_ICP    176554
Pb_ppm_AA_ICP    176036

HOLE ID          171003
SAMPFROM         171003
SAMPTO           171003
Soporte          171003
Cut_Pct          171071
Length: 70, dtype: int64
```

```
##To replace numeric missing values
df.fillna(df.mean(),inplace=True)
## The plot for missing values
```

```
sns.heatmap(df.isnull(),cbar=False,yticklabels=False,cmap='viridis')
```



Dropping the non-numeric due to much missing



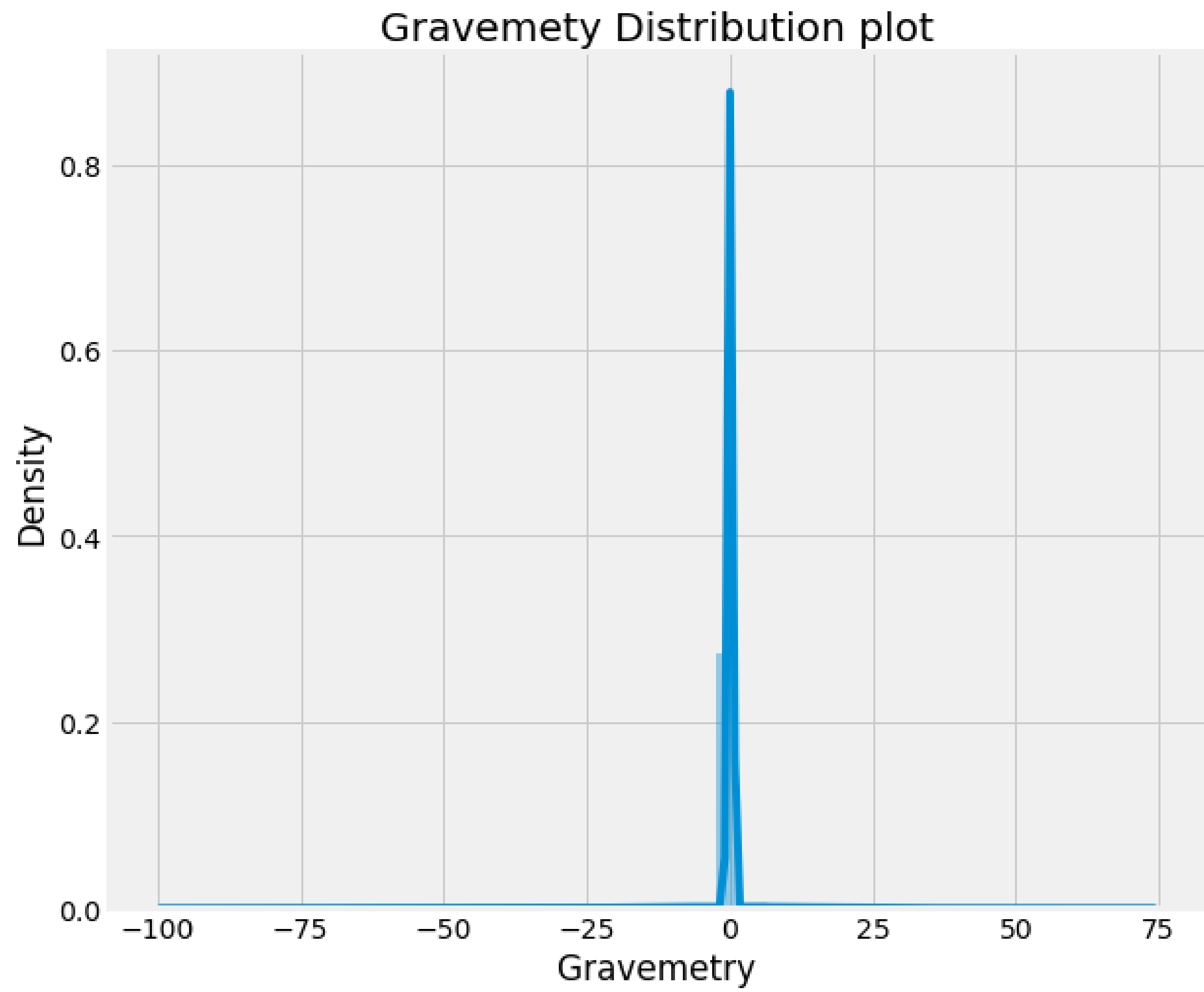
Plotting the distribution of Gravemetry

plt.figure(figsize=(10,8))

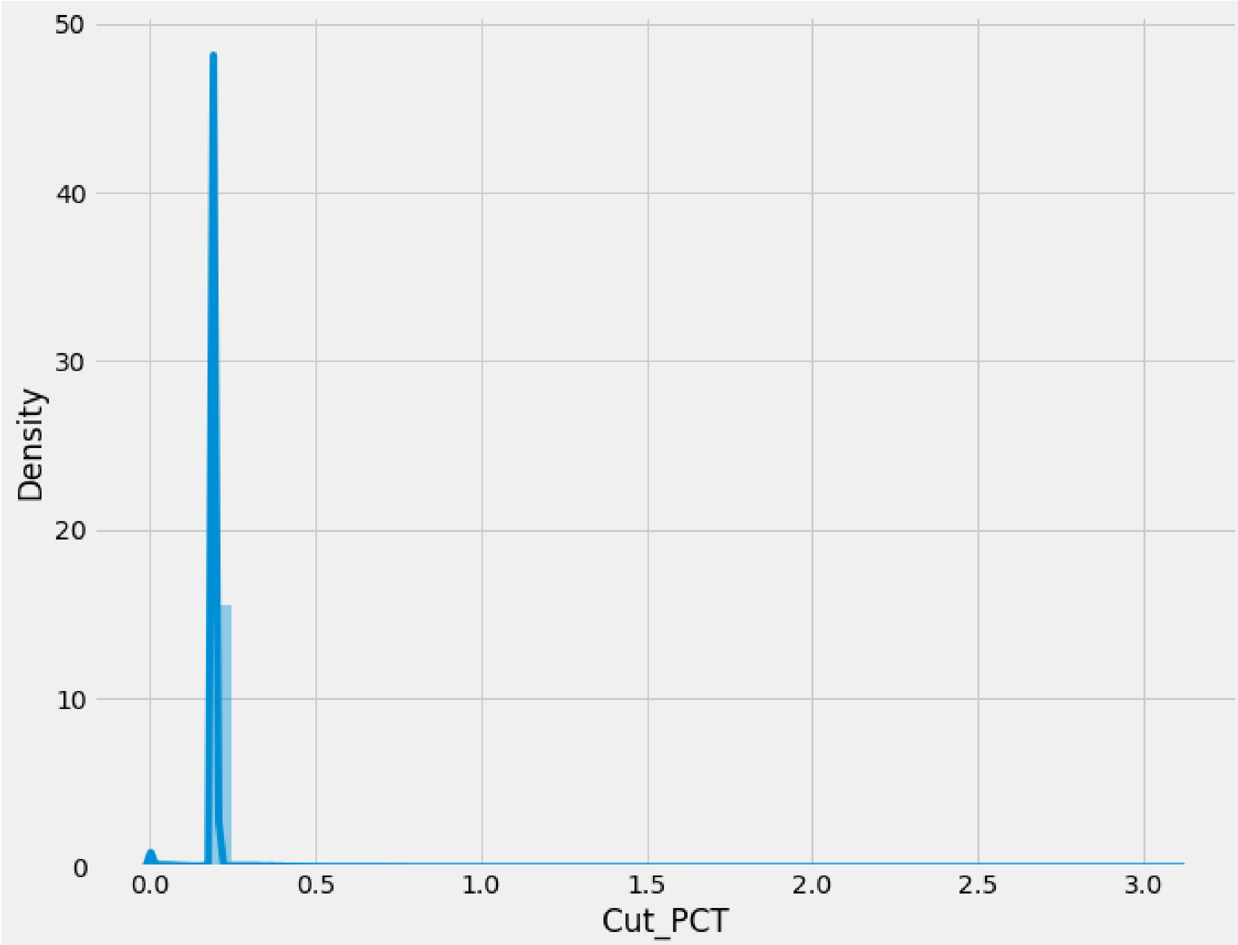
plt.style.use('fivethirtyeight')

sns.distplot(df['Gravemetry'])

plt.title("Gravemety Distribution plot")



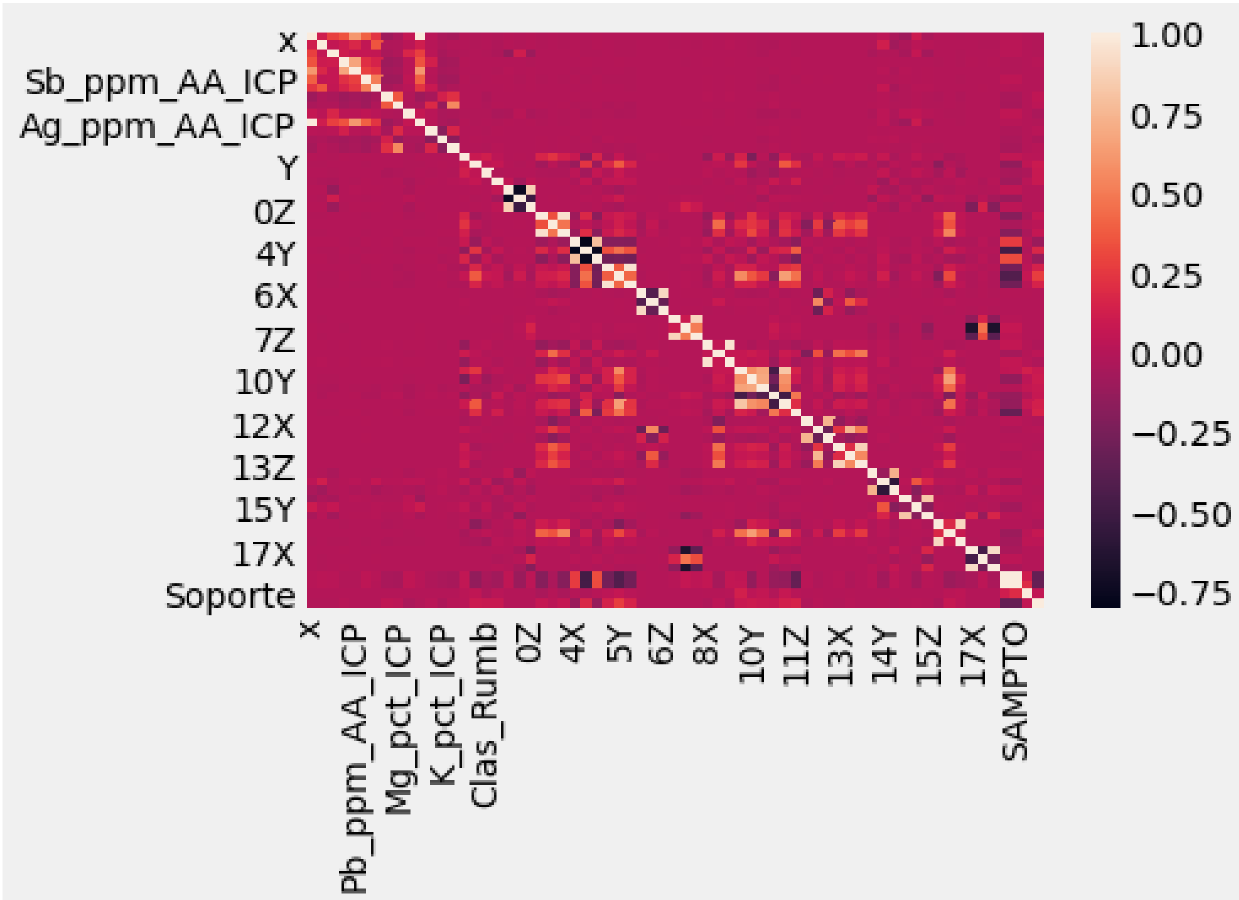

```
###

plt.figure(figsize=(10,8))

plt.style.use('fivethirtyeight')

sns.distplot(df['CuT_pct'])

plt.show()
```



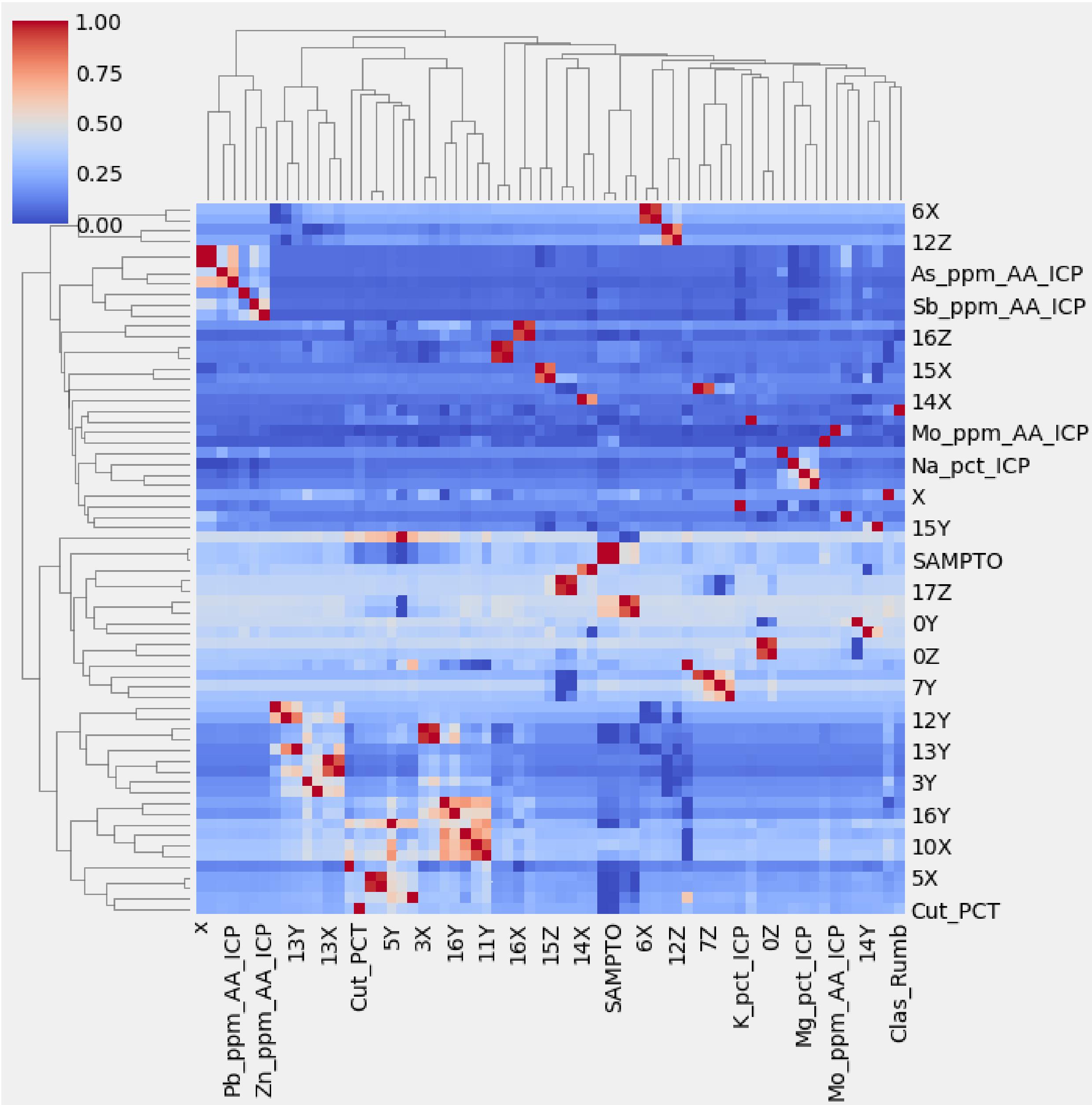
Relationship plot of all variables

Num_corr=df.corr()

sns.heatmap(Num_corr)


x
Sb_ppm_AA_ICP
Ag_ppm_AA_ICP
Y
0Z
4Y
6X
7Z
10Y
12X
13Z
15Y
17X
Soporte
x
Pb_ppm_AA_ICP
Mg_pct_ICP
K_pct_ICP
Clas_Rumb
0Z
4X
5Y
6Z
8X
10Y
11Z
13X
14Y
15Z
17X
SAMPTO
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Cluster Map



Normalizing the dataset

data_scaled = normalize(df)

data_scaled = pd.DataFrame(data_scaled, columns=df.columns)

data_scaled.head()

Out[45]:

	x	Au_ppb	Cu_ppm_AA_ICP	...	SAMPTO	Soporte	Cut_PCT
0	5.136584e-08	0.000002	1.797805e-06	...	0.000007	4.815548e-08	3.210365e-11

1 1.926214e-08 0.000002 2.568285e-07 ... 0.000007 4.815535e-08 3.210357e-11

2 1.284139e-08 0.000002 2.568279e-07 ... 0.000007 4.815523e-08 3.210348e-11

3 1.284136e-08 0.000002 1.605170e-07 ... 0.000007 4.815511e-08 3.210341e-11

4 1.926200e-08 0.000002 5.457567e-07 ... 0.000007 4.815500e-08 2.889300e-11

Machine Learning Regression

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, y_train)
```

##Printing intercept

```
lm.intercept_
-8.209566446935597e-08
```

##Printing the coefficient

```
              Coefficent
Ag_ppm_AA_ICP  1.085413e-04
Au_ppb        -7.813732e-06
Cu_ppm_AA_ICP  5.482249e-07
As_ppm_AA_ICP  1.092640e-05
Pb_ppm_AA_ICP -2.751776e-06
Sb_ppm_AA_ICP  3.965024e-05
Zn_ppm_AA_ICP  5.110316e-06
Na_pct_ICP     1.422139e-02
Mg_pct_ICP     9.138643e-03
Mo_ppm_AA_ICP -2.318827e-04
```

X 1.808953e-08

Y 3.377537e-07

Clas_Rumb 2.671339e-04

Gravemetry 6.423099e-04

##Prediction

model_output = pd.DataFrame(prediction,y_test, columns = ['Prediction'])

model_output.head()

Metrics

from sklearn import metrics

print('RMSE', np.sqrt(metrics.mean_squared_error(y_test,prediction)))

RMSE 1.6144876911655055e-09

R^2

metrics.explained_variance_score(y_test,prediction)

0.025012755908466522

Novustack's Mineral Forecast Team Recommendation and Suggestion

Recommendation

The importance of adopting data mining techniques to mining exploration cannot be overemphasized. With the increase in the quality and quantity of exploration data in the last 20 years, there is a need to discover new ways to apply techniques such as data visualization and probabilistic modeling to aid the discovery of economic deposits.

Due to the short time allocated to work on the data, we were able to come up with some results. Although, we believe that there are many more ways the data can be explored to achieve greater proximity to the objective (which is to develop a predictive model using M algorithms that easily identify prospective targets for mining exploration).

Multiple linear regression was the statistical tool of resort as it uses several explanatory variables to predict the outcome of a response variable. It is an extension of linear regression. Although, this method has its limitation in cases of data incompleteness.

Researching for suitable algorithms that could proffer reliable results, the following discoveries were made which could in turn aid the decision-making process on which algorithm is most appropriate.

According to [1], using a dataset with known mineral deposits' locations, exploration characteristics can be selected by applying probabilistic modeling to predict the probability estimates of discovering other possible economic deposits. Although this approach will provide useful results, its shortcomings are a result of the basic assumptions of its weights of evidence. But, a method like randomForest has proven to be effective due to the algorithm's random selection scenarios of both sample and variable, reducing the correlation among the individual decision trees, increasing the diversity of the forest avoiding overfitting. Algorithms like support vector machine (SVM) works best when there is a binary classification (mineral occurrence or non-occurrence) based on careful selection of the data inputs.

Research articles have shown that neural network and kernel methods are most widely used algorithms for data related to mining exploration due to the enormous benefits it offers which include:

The back-propagation procedure of ANN computes the output errors between the predicted value and real target value which is fed back to the ANN, and then ANN tunes the weights and biases in the network to minimize the output error

References

1. Barnett, C. T., and Williams, P. M., 2006, Mineral exploration using modern data mining techniques, in Doggett, M. E. and Parry, J. R. (eds.), Wealth Creation in the Minerals Industry: Integrating Science, Business and Education. Society of Economic Geologists, Special Publication 12, pp. 295-310.
2. https://www.bwmining.com/casestudies
3. https://minervaintelligence.com/economic-geology/
4. https://goldspot.ca/