# Machine Learning Example

ML is process of finding a mapping function by analysing the provided data with the help of ML algorithm and making final prediction baesd on that found mapping function.

# ML Steps

1. Data construction
2. Data Pre-processing ( i. analysis ii. feature correlation iii. feature selection iv. feature scaling)
3. Hyper-parameter Optimization
4. Model tuning
5. Model evaluation

# Lets begin

## Importing libraries

```
#Load necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, m
```

# Loading data

```
from google.colab import drive
drive.mount('/content/gdrive')

xl_file = '/content/gdrive/My Drive/Colab Notebooks/Metal_hallide.xlsx'
csv_file = '/content/gdrive/My Drive/Colab Notebooks/Metal_hallide_data.csv'
data = pd.read_excel(xl_file)
data.to_csv(csv_file, index = False)
print(f'Conversion complete. Saved as {csv_file}')
```

```
#This is not necessary in this case ie. for google colab
#xl_file = 'half_heusler_data.xlsx'
#csv_file = 'half_heusler_data.csv'
#data = pd.read_excel(xl)
#data.to_csv(csv_file,index=False)
#print(f"Excel file is sucessfully converted, Saved as {csv_file}")

#uploading converted csv_file
data = pd.read_csv(csv_file)
data
```

| | Compound(xyz) | r x(Å) | ry(Å) | rz(Å) | a(Å) | ez |
|---|---|---|---|---|---|---|
| **0** | CoMnP | 0.90 | 0.96 | 0.80 | 5.34 | 2.19 |
| **1** | CoFeP | 0.90 | 0.78 | 0.80 | 5.35 | 2.19 |
| **2** | CoMnSi | 0.90 | 0.96 | 0.85 | 5.36 | 1.90 |
| **3** | CoVP | 0.90 | 0.72 | 0.80 | 5.36 | 2.19 |
| **4** | CoCrSi | 0.90 | 0.57 | 0.85 | 5.36 | 1.90 |
| **...** | ... | ... | ... | ... | ... | ... |
| **132** | FeMnSi | 0.78 | 0.96 | 0.85 | 5.32 | 1.90 |
| **133** | FeVP | 0.78 | 0.72 | 0.80 | 5.31 | 2.19 |
| **134** | FeFeP | 0.78 | 0.78 | 0.80 | 5.31 | 2.19 |
| **135** | MnCrP | 0.96 | 0.57 | 0.80 | 5.30 | 2.19 |
| **136** | FeCrP | 0.78 | 0.57 | 0.80 | 5.29 | 2.19 |

137 rows × 6 columns

## Data Preprocessing

```
Analysing and handling missing data
Data visualization, removing outliers
Encoding catagorical values
Feature scaling
Feature engineering
```

```
#data range
data.index

#to know about data types
data.info()
#data.dtypes
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 137 entries, 0 to 136
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Compound(xyz)  137 non-null    object 
 1   r x(Å)         137 non-null    float64
 2   ry(Å)          137 non-null    float64
 3   rz(Å)          137 non-null    float64
 4   a(Å)           137 non-null    float64
 5   ez             137 non-null    float64
dtypes: float64(5), object(1)
memory usage: 6.5+ KB
```

```
data.isnull().sum(axis=0)
```

```
#knowing the vaccant row or column in data
#data.isnull().sum()
data.isnull().sum(axis=0) #along rows for each column
#data.isnull().sum(axis=1)#along columns for each row

#Finding along columns
null_columns = data.columns[data.isnull().any()]
print(null_columns)

#Printing rows of that particular column
null_values = data[data['a(Å)'].isnull()]
null_index = null_values.index
#print(null_values)
print(null_index)

#finding along rows
null_rows = data[data.isnull().any(axis=1)]
print(null_rows)

data.duplicated().any()
```

```
duplicate = data[data.duplicated()]
print(duplicate)

#finding non duplicate data
data.nunique()
```

```
Compound(xyz)    136
r x(Å)             7
ry(Å)              6
rz(Å)              6
a(Å)              71
ez                 9
dtype: int64
```

```
data.columns
```

```
data[['Compound(xyz)','a(Å)']]
```

```
#knowing tha value at apticular position
data['ez'].loc[110]
```

```
#printing rows
data.loc[136]
```

```
#matching the values
data[data['ez']==2.19]
```

```
data[data['ez']<2.19]
```

```
#printing columns that corresponds to maximum value of another column
data[['a(Å)','ez']][data.ez==data['ez'].max()]
```

```
#printinng statistical parameter
data.ez.mean()
```

```
data.ez.max()
```

```
data.describe()
```

```
#Visulising data
# 1. Histogram
import seaborn as sns
sns.set_style('whitegrid')
sns.histplot(data['a(Å)'], bins = 20, color='green',edgecolor='black')
plt.xlabel("Lattice parameter")
plt.ylabel("frequency")

plt.show()
```

```
data.boxplot()
plt.show()
```

```
sns.heatmap(data.corr(),annot=True,cmap='coolwarm',fmt='.2f',linewidths=0.5)
plt.show()
```

## Feature engineering

### 1. Labeling features and target

```
features = ['r x(Å)', 'ry(Å)', 'rz(Å)', 'a(Å)',]
target = data['ez']
```

```
X = data[features]
y = target
```

```
y = target
print("shape of X = ",X.shape)
print("shape of y = ",y.shape)
```

## 2. Feature Selection

Random forest algorith is generally used to select proper features for the provided target variables.

## 2. Feature Scalig

### Standard Scaler()

```
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2,random_stat

print("shape of X_train = ",X_train.shape)
print("shape of X_test = ",X_test.shape)
#print("shape of y_train = ",y_train.shape)
#print("shape of y_test = ",X_test.shape)
```

```
sc = StandardScaler()
sc.fit(X_train)

#sc.mean_
#sc.scale_
```

```
X_train_sc = sc.transform(X_train)
X_test_sc = sc.transform(X_test)

X_train_sc = pd.DataFrame(X_train_sc, columns = ['r x(Å)', 'ry(Å)', 'rz(Å)', 'a
X_test_sc = pd.DataFrame(X_test_sc,columns=['r x(Å)', 'ry(Å)', 'rz(Å)', 'a(Å)']

X_test_sc
```

```
X_train_sc.describe().round(2)
```

```
sns.pairplot(X_train)
```

```
sns.pairplot(X_train_sc)
```

Hyper parameter optimization

Hyper-parameter optimization

Gridsearch CV, RandomizedSearch CV, Bayesian Optimization etc. But, in this case we are taking default values for tutorial purpose. Generally we train test all the models by taking default values and then compare the results to get highly predictive model and then we select those models and perform optimization and then further model tuning.

## Model Tuning

# Linear regression model

```
model_lr = LinearRegression()
model_lr.fit(X_train_sc, y_train)
```

## Making prediction

```
y_pred = model_lr.predict(X_test_sc)
y_pred_train = model_lr.predict(X_train_sc)
print(y_pred)

print(y_pred_train)
```

## Model evaluation

```
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test,y_pred)
r2 = r2_score(y_test,y_pred)

print(f" The MSE = {mse}")
print(f" The RMSE = {rmse}")
print(f" The R2 score = {r2}")
print(f" The MAE = {mae}")

new_features = [1.3,1.8, 2.3, 5.4]
new_ez = model_lr.predict([new_features])
print(new_ez)
```

## You can also perform k-Fold validation to validate your result

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import cross_val_score
mse_scorer = make_scorer(mean_squared_error)
mse_scores = cross_val_score(model_lr, X_train_sc, y_train, cv=5, scoring=mse_s
r2_scorer = make_scorer(r2_score)
r2_scores = cross_val_score(model_lr,X_train_sc, y_train, cv =5, scoring=r2_scc
print("Mean-squared error :", mse_scores)
print("R2 score :", r2_scores)
```

## Now test other regression model

```
Ridge, Kernel Ridge Regressor, Lasso, DecessionTree regressor, Random forest regressor,
```

## Data Visualisation

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.7, label='Actual vs Predictec
plt.scatter(y_train, y_pred_train, color='green', alpha=0.7, label='Actual vs F
plt.plot(y_test, y_test, color='red', label='Perfect Prediction Line')
plt.title('Linear Regression: Actual vs Predicted for ez')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```