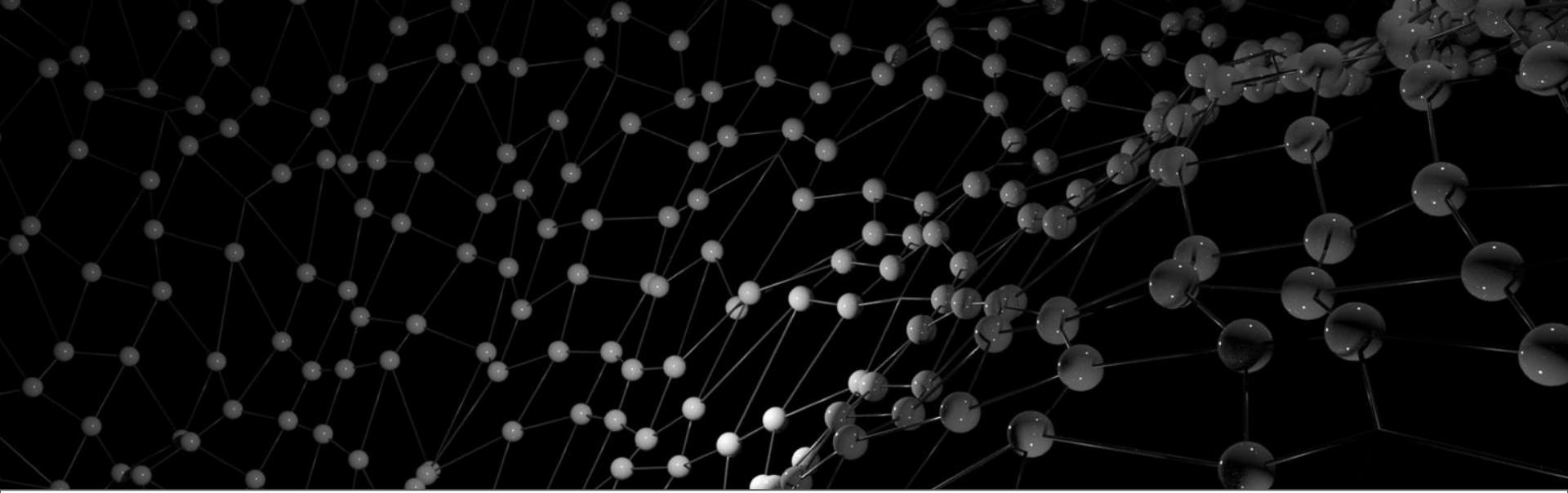
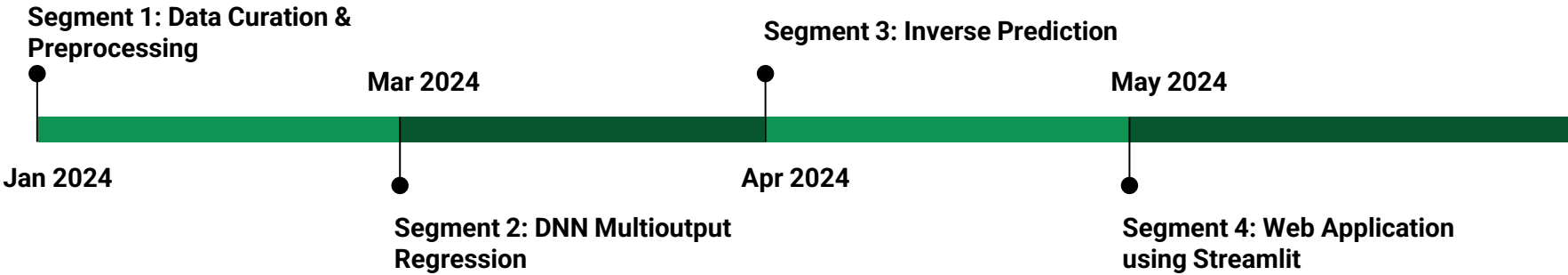


Accelerated Design for High Entropy Alloys Based on DNN-Multioutput Regression

Institute of High Performance Computing
A*STAR IHPC

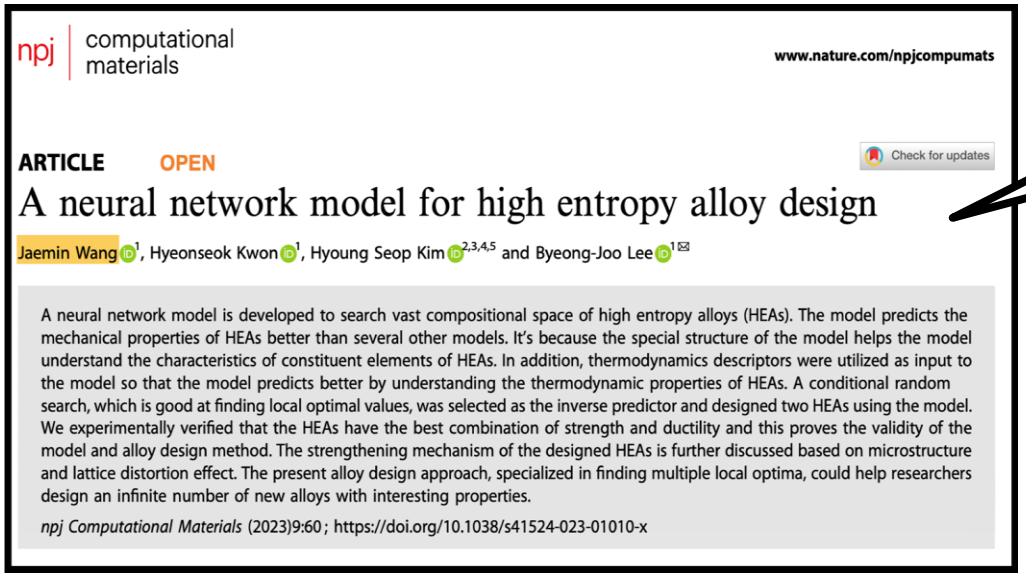
A*STAR ARIA Research Internship Project 2024
Wong Qi Yuan , Jeffrey (NUS, B.Sc (Hons) in Statistics)
Main Supervisor: Dr. Leong ZhiDong

Project Timelines



Main Reference

- The main study primarily focuses on developing and validating a neural network model for predicting the mechanical properties of high entropy alloys.



npj computational materials

www.nature.com/npjcompumats

ARTICLE OPEN

Check for updates

A neural network model for high entropy alloy design

Jaemin Wang[1], Hyeonseok Kwon[1], Hyoung Seop Kim[2,3,4,5] and Byeong-Joo Lee[1]✉

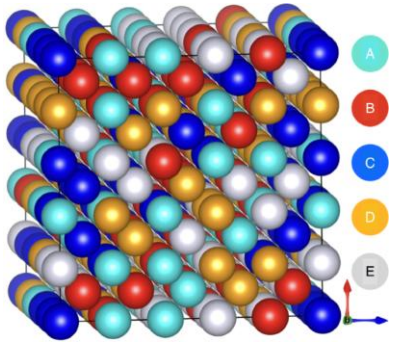
A neural network model is developed to search vast compositional space of high entropy alloys (HEAs). The model predicts the mechanical properties of HEAs better than several other models. It's because the special structure of the model helps the model understand the characteristics of constituent elements of HEAs. In addition, thermodynamics descriptors were utilized as input to the model so that the model predicts better by understanding the thermodynamic properties of HEAs. A conditional random search, which is good at finding local optimal values, was selected as the inverse predictor and designed two HEAs using the model. We experimentally verified that the HEAs have the best combination of strength and ductility and this proves the validity of the model and alloy design method. The strengthening mechanism of the designed HEAs is further discussed based on microstructure and lattice distortion effect. The present alloy design approach, specialized in finding multiple local optima, could help researchers design an infinite number of new alloys with interesting properties.

npj Computational Materials (2023)9:60 ; https://doi.org/10.1038/s41524-023-01010-x

"... modeling elongation is much more difficult than modeling strength. This is because the mechanism for enhancing elongation is so complex…"

https://doi.org/10.1038/s41524-023-01010-x

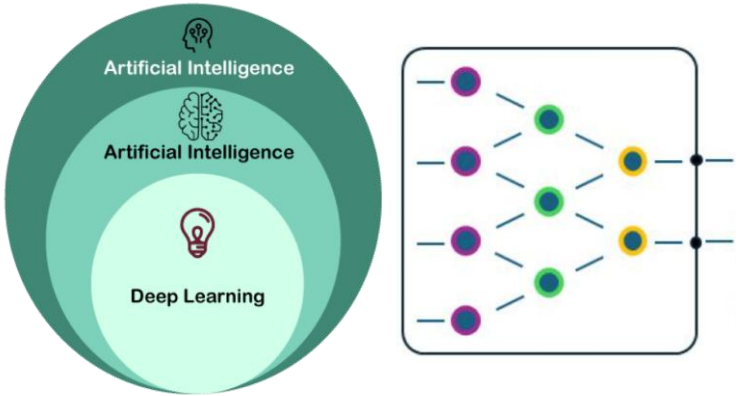
What is Deep Learning for High Entropy Alloys?



High Entropy Alloys

- A new class of materials composed of multiple principal elements - contributes to exceptional mechanical properties.
- The vast compositional space of HEAs makes traditional experimental approaches to material discovery and optimisation time-consuming and costly.

- Deep learning is a subset of machine learning that uses neural networks to understand complex patterns in the data.
- By leveraging deep learning models, we can rapidly analyze and predict the properties of HEAs based on their processing conditions and other properties.



Segment 1: Data Preprocessing

Objective: Organising and Managing a Collection of Dataset

Structure of Tabular Data

C	Al	V	Cr
Mn	Fe	Co	Ni
Cu	Mo	Ti	B

12 Elements

Unique HEA

Chemical Compositions (%) : Used to create Matminer Elemental Descriptors, but not used for model training.

Processing Parameters : Used for model training

Matminer Elemental Descriptors: Derived from Matminer Library based on chemical compositions and used for model training

Experimental Variables: Used for model training

	C	: : :	B	homo_temp	cold_rolling	anneal_temp	anneal_time	Density	: : :	constraint_1	YS (MPa)	UTS (MPa)	EL (%)
1													
2													
3													
:													
:													
:													
:													
:													
568													
569													

Temperature at which HEA is heated to make its composition uniform and eliminate chemical segregation

Passed through rollers at R.T to reduce its thickness and improve its mechanical properties

To relieve internal stresses and improve ductility and toughness

To achieve the desired changes in its microstructure and mechanical properties

To adjust for a realistic baseline annealing temperature and to capture the interaction between anneal-time and anneal-temp.
Formula: (anneal-temp - 300) x anneal_time

20	Density	569 non-null	float64
21	Miedema_dH_inter	569 non-null	float64
22	Miedema_dH_amor	569 non-null	float64
23	Miedema_dH_ss_min	569 non-null	float64
24	Yang delta	569 non-null	float64
25	Yang omega	569 non-null	float64
26	APE mean	569 non-null	float64
27	Radii local mismatch	569 non-null	float64
28	Radii gamma	569 non-null	float64
29	Configuration entropy	569 non-null	float64
30	Lambda entropy	569 non-null	float64
31	Electronegativity delta	569 non-null	float64
32	Electronegativity local mismatch	569 non-null	float64
33	VEC mean	569 non-null	float64
34	Mixing enthalpy	569 non-null	float64
35	Mean cohesive energy	569 non-null	float64
36	Shear modulus mean	569 non-null	float64
37	Shear modulus delta	569 non-null	float64
38	Shear modulus local mismatch	569 non-null	float64
39	Shear modulus strength model	569 non-null	float64
40	mean atomic_mass	569 non-null	float64
41	std_dev atomic_mass	569 non-null	float64
42	mean atomic_radius	569 non-null	float64
43	std_dev atomic_radius	569 non-null	float64
44	mean mendeleev_no	569 non-null	float64
45	std_dev mendeleev_no	569 non-null	float64
46	mean electrical_resistivity	569 non-null	float64
47	std_dev electrical_resistivity	569 non-null	float64
48	mean thermal_conductivity	569 non-null	float64

49	std_dev thermal_conductivity	569 non-null	float64
50	mean melting_point	569 non-null	float64
51	std_dev melting_point	569 non-null	float64
52	mean bulk_modulus	569 non-null	float64
53	std_dev bulk_modulus	569 non-null	float64
54	mean coefficient_of_linear_thermal_expansion	569 non-null	float64
55	std_dev coefficient_of_linear_thermal_expansion	569 non-null	float64

Data Extraction (Manual)

Chemical Compositions

2. Experimental

2.1. Fabrication of $V_{20}Cr_{15}Fe_{20}Ni_{45}$ HEA

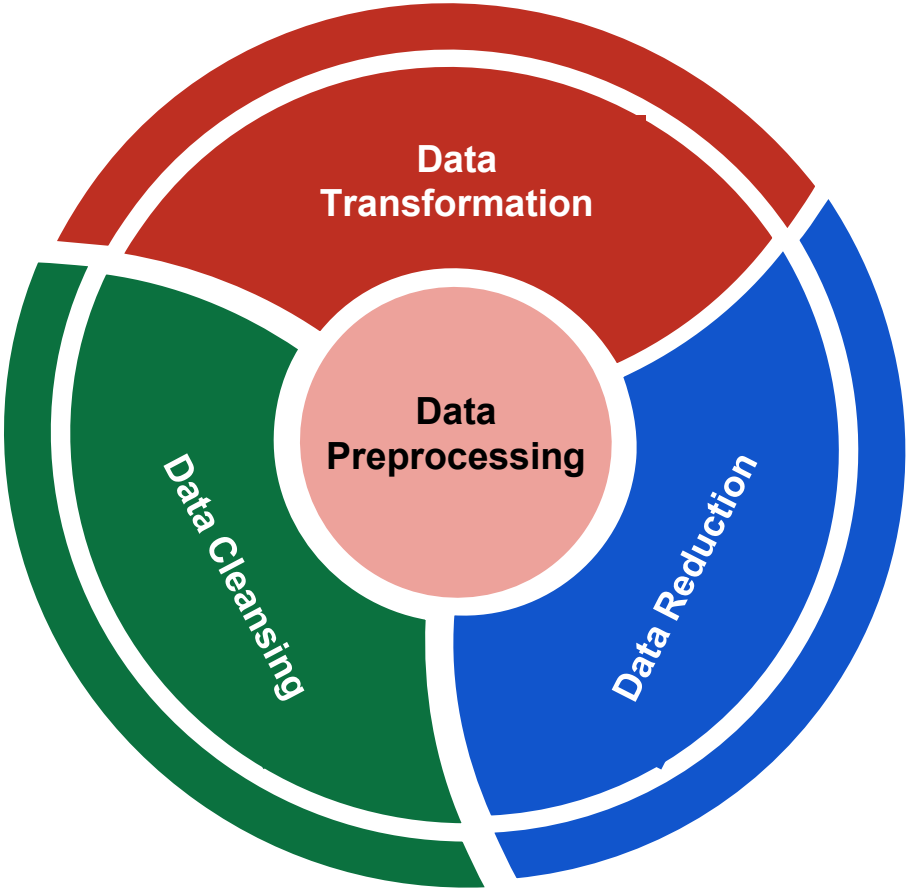
The HEA was fabricated by a compact vacuum induction melting equipment (model; MC100V, Indutherm, Walzbachtal-Wossingen, Germany) under an argon atmosphere using commercial pure elements (the purity of each raw material was 99.9% at least). Raw elements were alloyed in a ZrO_2-coated alumina crucible to make a master alloy of 150 g in weight. The master alloy was molten and poured into a rectangular graphite module (length; 100 mm, width; 35 mm, thickness; 8 mm). The ingot was homogenized at 1100 °C for 6 h, pickled in a 20% HCl, and then cold-rolled (75%) at room temperature with a thickness reduction of 75% to produce a 2-mm-thick sheet. The sheets were annealed at 1000 °C, 950 °C, 900 °C for 10 min. Since the annealing time for 10 min at 800 °C is not enough for the full recrystallization of the cold-rolled HEA sheet (Supplementary Fig. S1), the sheet was annealed for 1 h at that temperature. For convenience, the specimens annealed at 1000, 950, 900, and 800 °C are referred to as 1000A, 950A, 900A, and 800A, respectively.

Processing Parameters

Table 2
Room- and cryogenic tensile properties of the annealed $V_{20}Cr_{15}Fe_{20}Ni_{45}$ HEA specimens.

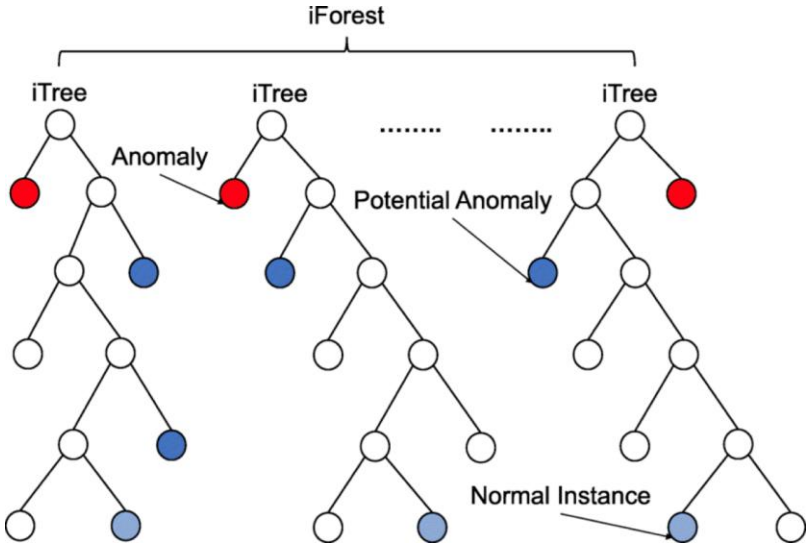
Testing temperature	HEA Specimen	Yield strength (MPa)	Tensile strength (MPa)	Elongation (%)
25 °C	1000A	352 ± 3	790 ± 4	64.2 ± 3.0
	950A	555 ± 4	976 ± 4	45.6 ± 2.2
	900A	587 ± 9	1013 ± 8	38.5 ± 0.4
	800A	715 ± 13	1149 ± 7	19.6 ± 4.0
− 196 °C	1000A	571 ± 7	1119 ± 3	73.3 ± 1.2
	950A	794 ± 2	1326 ± 6	47.4 ± 0.9
	900A	819 ± 13	1369 ± 7	42.7 ± 2.1
	800A	933 ± 6	1493 ± 8	22.5 ± 2.2

Experimental Variables


Data Transformation
Data Preprocessing
Data Cleansing
Data Reduction

01	Drop Chemical Compositions	• Not benefit to the neural networks.
02	Check for Missing Values	• To ensure data completeness and prevent bias in model training.
03	Renaming Columns	• To improve readability and consistency
04	Check for Duplicate Rows	• To ensure data integrity and prevent bias in model training.
05	Removing Potential Outliers	• To enhance the quality and performance of the data models.

Isolation Forest



- **Isolation Forest (iForest)** constructs random trees with random splits.
- In high-dimensional data, anomalies are isolated in fewer splits.
- **Anomalies** → shorter path lengths → negative score
- **Normal** → longer path lengths → positive score

Isolation Forest

```
# define features to consider for anomalies detection

# create Isolation Forest model
model = IsolationForest(n_estimators = 60, random_state=42)

# fit the model to the data
model.fit(hea_ed_df_copy[features_names])

# get anomalies scores for each data point
anomalies_scores = model.decision_function(hea_ed_df_copy[features_names])

# make a copy to store the anomalies scores
hea_ed_df_scores = hea_ed_df_copy.copy()

# store the scores into the dataframe
hea_ed_df_scores['anomalies_scorers'] = anomalies_scores
hea_ed_df_scores
```

A code snippet for Anomalies Detection using Isolation Forest

- **n_estimators = 60** means that the iForest will build 60 isolation trees.
- Each tree randomly selects a data subset and makes random splits.

```
# setting threshold to identify anomalies (adjust as needed)
threshold = 0.0

# identify and display the anomalies rows
hea_ed_df_scores[anomalies_scores < threshold]
```

A code snippet to display the anomalies rows

Isolation Forest

an_thermal_conductivity	mean_melting_point	std_dev_melting_point	mean_coef_linear_thermal_expansion	yield_strength	ultimate_tensile_strength	elongation	anomalies_scorers
128.740000	1595.254000	181.489276	0.000016	235.0	585.0	60.4	-0.000067
89.320000	1913.630000	436.513853	0.000011	617.0	1057.0	52.0	-0.032262
90.333333	1769.000000	41.509035	0.000013	195.0	507.0	32.0	-0.102129
85.500000	1769.500000	58.689863	0.000013	180.0	500.0	36.0	-0.045288
95.500000	1748.000000	28.284271	0.000013	100.0	535.0	42.0	-0.124854
...	...	...	...	...	...	...	...
144.573125	1663.224196	454.230100	0.000016	1018.0	1200.0	11.0	-0.092646
144.573125	1663.224196	454.230100	0.000016	915.0	1085.0	19.5	-0.091172
144.573125	1663.224196	454.230100	0.000016	617.0	915.0	46.0	-0.091839
144.573125	1663.224196	454.230100	0.000016	405.0	810.0	60.5	-0.095997
144.573125	1663.224196	454.230100	0.000016	1240.0	1400.0	5.1	-0.092289

A partial output of anomalies rows

Spearman Correlation Test

- Spearman Correlation is robust to non-linear relationships and non-normality.
- Unlike Pearson, it is less sensitive to outliers and non-normal distributions.
- Choosing p-value = 0.1 balances identifying potential relationships and minimising false positives.

```
# call the function to get significant features
significantFeatures = get_significant_features(x_ed, y, 0.1)
# display the significant feature names
significantFeatures
```

A code snippet to call the function to perform Spearman Correlation Test for Elemental Descriptors

```
# define a function to get significant features based on Spearman correlation p-values
def get_significant_features(x, y, significance_level):
    # extract feature names
    feature_names = x.columns
    # define empty list to store significant features and p-values
    significant_features = []
    p_val_ys = []
    p_val_uts = []
    p_val_el = []


    # loop through each feature
    for name in feature_names:
        # calculate the Spearman correlation and p-value with yield strength target variable
        corr1, p_val1 = spearmanr(x[name], y['yield_strength'])
        # calculate the Spearman correlation and p-value with ultimate tensile strength target variable
        corr2, p_val2 = spearmanr(x[name], y['ultimate_tensile_strength'])
        # calculate the Spearman correlation and p-value with elongation target variable
        corr3, p_val3 = spearmanr(x[name], y['elongation'])

        # check all p-values are less than or equal to the significance level
        if p_val1 <= significance_level and p_val2 <= significance_level and p_val3 <= significance_level:
            # if so, consider the feature significant and add it to the list and the p-values
            significant_features.append(name)
            p_val_ys.append(p_val1)
            p_val_uts.append(p_val2)
            p_val_el.append(p_val3)
        overall_significant_features = pd.DataFrame({"significant_features": significant_features,
                                                     "p-value (YS)": p_val_ys,
                                                     "p-value (UTS)": p_val_uts,
                                                     "p-value (EL)": p_val_el})
    return overall_significant_features
```

Screenshot

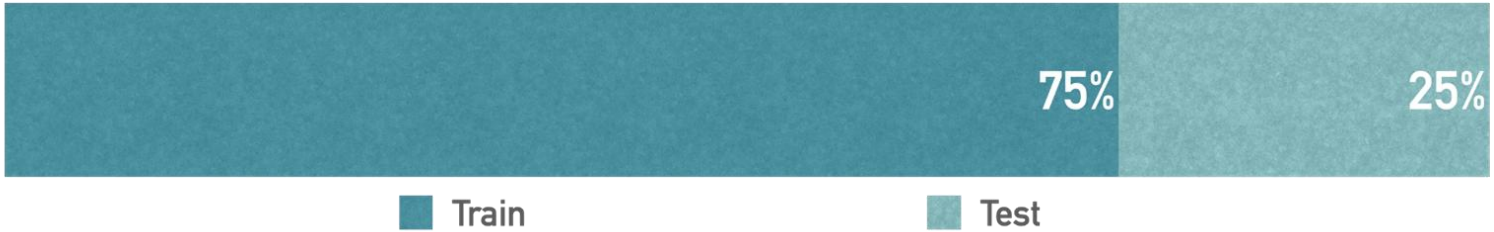
A code snippet to perform Spearman Correlation Test for Elemental Descriptors

Spearman Correlation Test

	significant_features	p-value (YS)	p-value (UTS)	p-value (EL)
0	deltaH_ss_min	3.627014e-04	2.026764e-04	1.789746e-06
1	yang_delta	6.280654e-04	9.142199e-05	4.501745e-05
2	yang_omega	6.728885e-04	5.533649e-06	8.769394e-02
3	Radii local mismatch	2.808766e-04	3.144936e-04	8.928174e-10
4	radii_gamma	4.633767e-02	5.649267e-02	7.375255e-02
5	lambda_entropy	2.765629e-03	2.270585e-04	5.144087e-04
6	mixing_enthalpy	1.139413e-05	3.173295e-07	2.548260e-03
7	mean_cohesive_energy	2.499590e-04	7.407353e-09	4.794176e-03
8	shear_modulus_mean	2.612888e-02	2.481845e-04	2.124662e-06
9	shear_modulus_delta	8.383964e-09	3.522741e-09	4.099954e-09
10	shear_modulus_local_mismatch	1.415289e-12	3.241322e-13	1.960834e-08
11	Shear modulus strength model	1.317759e-09	9.734565e-10	6.252160e-10
12	std_dev_atomic_mass	2.962282e-11	8.731977e-16	2.066158e-05
13	mean_atomic_radius	6.337249e-06	7.106851e-08	2.653432e-03
14	std_dev_atomic_radius	1.169817e-04	1.220897e-06	9.898154e-04
15	mean_mendeleev_no	7.630512e-03	3.232058e-02	3.434432e-08
16	std_dev_mendeleev_no	2.209310e-07	8.887338e-08	2.824610e-09
17	mean_thermal_conductivity	2.930930e-07	1.560080e-07	2.021477e-05
18	mean_melting_point	5.620587e-04	4.919313e-08	8.406982e-07
19	std_dev_melting_point	4.361927e-12	6.335302e-15	6.336939e-07
20	mean_coef_linear_thermal_expansion	3.453755e-05	1.303918e-09	3.172723e-04

Results for Elemental Descriptors that show statistical significant relationships (p-val <= 0.1) with YS, UTS, and EL

Train and Test Data Creation




```
# display the shape of the data splitting
print(X_train_ed.shape)
print(X_test_ed.shape)
print(y_train_ed.shape)
print(y_test_ed.shape)
```



```
(373, 26)
(125, 26)
(373, 3)
(125, 3)
```

Data Transformation

Goal: To make sure features are on almost the same scale so that each feature is equally important and make it easier to process by DNN model.

Data Transformation

Standard Scaler

$$x_i^{'} = \frac{x_i - \mu}{\sigma}$$

- Standardizes all input features.
- Can be helpful if the data follows a normal distribution, but this does not have a necessarily to be true.
- If you have outliers in your data, they will not be affected by standardization.

Min-Max Normalization

$$x_i^{'} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

- It does not assume any distributions.
- Scales the target variables to a specified range of [0, 1].
- To ensure positive predicted target variables.

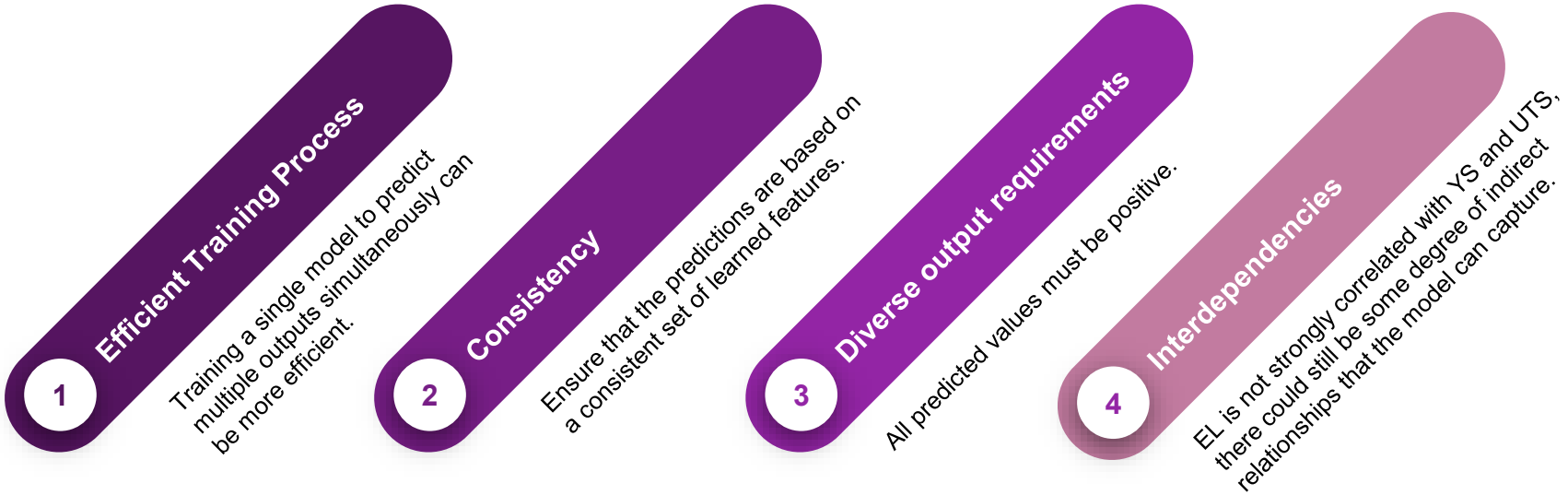
	homo_temp	cold_rolling	anneal_temp	anneal_time	constraint_1	deltaH_ss_min	yang_delta	yang_omega	Radii local mismatch	radii_gamma	..
234	0.438615	-0.144903	0.308334	-0.081506	-0.117422	-0.912638	-0.703833	-0.027736	-0.282347	-0.554455	..
348	-1.947512	0.997238	-1.418179	-0.280852	-0.307648	0.362279	0.392238	-0.210813	1.118882	-0.115773	..
42	0.612405	-0.144903	0.308334	-0.247628	-0.275944	-0.757946	-0.703833	0.036508	-0.282347	-0.554455	..
447	0.612405	0.283400	0.588916	-0.081506	-0.071622	-0.912638	-0.703833	-0.027736	-0.282347	-0.554455	..
255	-1.947512	0.997238	-1.418179	-0.280852	-0.307648	2.137844	1.742925	-0.346089	2.408301	-0.116270	..
...	...	...	...	...	...	...	...	...	...	...	..
403	0.612405	0.426167	0.588916	-0.247627	-0.268310	-0.746567	-0.765483	0.133720	-0.545696	-0.554030	..
329	0.612405	0.140632	0.775970	0.117839	0.225473	0.014174	-0.583148	-0.187952	-0.542569	-0.211099	..
413	0.438615	0.711702	-1.418179	-0.280852	-0.307648	0.695257	0.034109	-0.196139	-0.172630	-0.112747	..
48	0.264826	0.711702	1.150079	-0.181179	-0.143834	-0.396755	1.922461	-0.165282	0.880466	2.668364	..
230	0.612405	0.671728	0.963024	-0.231015	-0.233375	1.329378	0.912708	-0.261798	1.801616	-0.116340	..

	yield_strength	ultimate_tensile_strength	elongation
234	0.299943	0.340537	0.529843
348	0.655348	0.692207	0.084817
42	0.365025	0.304519	0.166492
447	0.121675	0.199083	0.547644
255	0.865874	0.959398	0.061780
...	...	...	...
403	0.189587	0.283563	0.704712
329	0.209394	0.377865	0.737173
413	0.565365	0.569745	0.099476
48	0.161290	0.311067	0.690052
230	0.359366	0.414538	0.303665

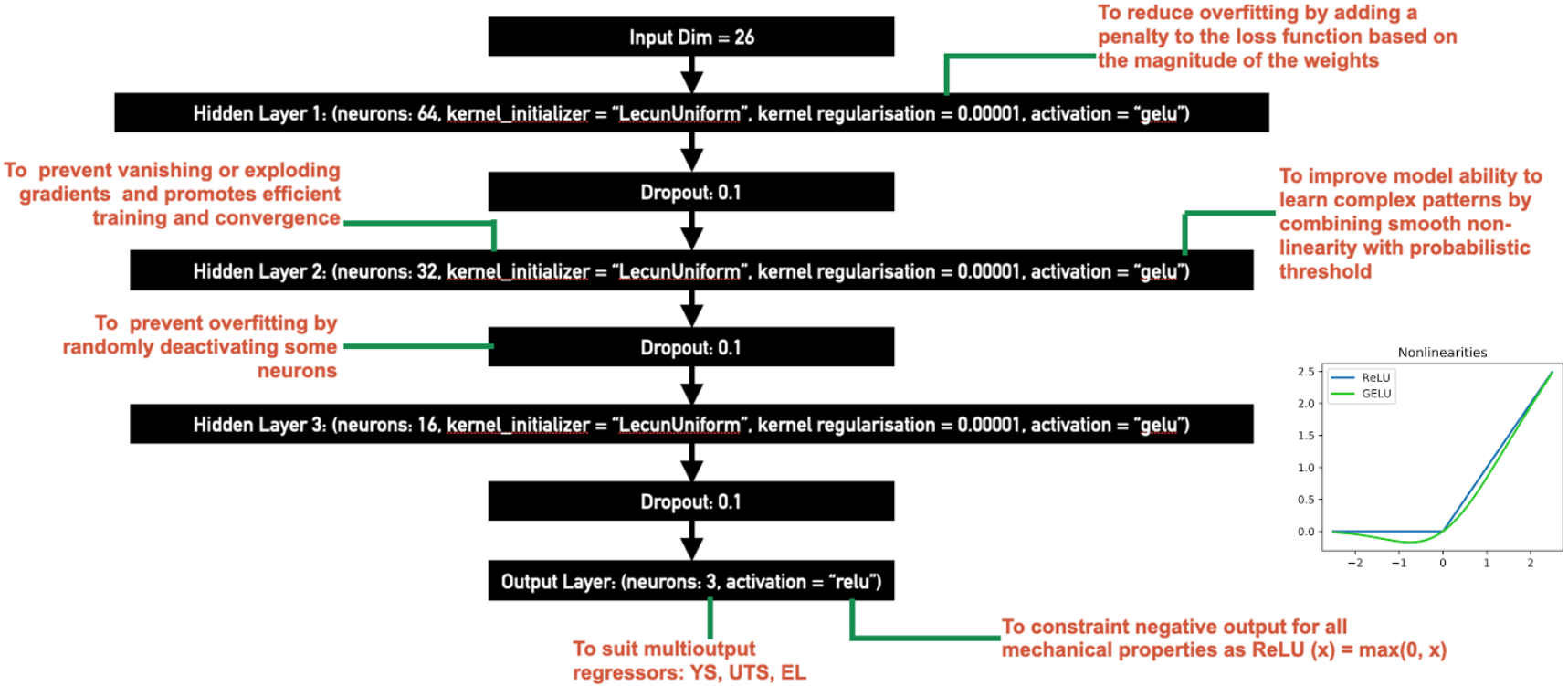
Segment 2: DNN-Multioutput Regression

Objective: To provide actionable insights and predictions

Reasons for choosing DNN-Multioutput Regression



DNN Multioutput Regression Architecture



DNN Multioutput Regression Architecture

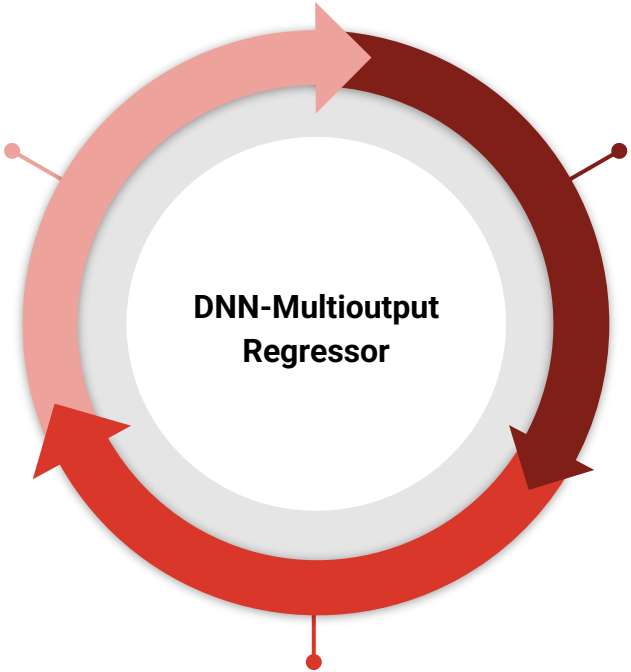
```
# define base model
dnn_gs_ed_cv = KerasRegressor(model = dnn_gs_mo,
                              loss = custom_mse_loss,
                              batch_size = 32,
                              epochs = 300,
                              optimizer = Adamax(learning_rate = 0.01),
                              metrics = [R2Score, RootMeanSquaredError],
                              random_state = 42)
```

A code snippet for Keras-Regressor Wrapper

Model Training

Forward Propagation

To predict all target variables simultaneously by taking the input, calculating weighted sum, adding bias to it, applying activation function, and then passing the results to the next layers in the neural networks.



Compute MSE Loss

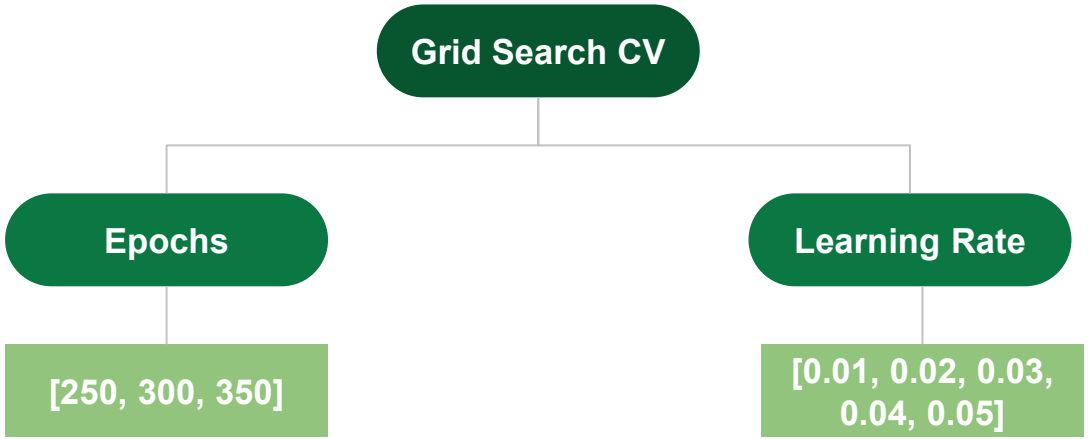
Averaging the squared differences between the predicted and actual values across all target variables.

Backward Propagation

Differentiating the loss w.r.t weights and biases and then adjusting to these parameters using an optimization algorithm.

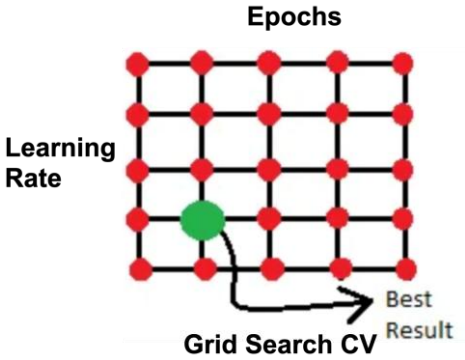
Loss	Mean Square Error
Batch Size	32
Epochs	300
Optimizer	Adamax
Learning Rate	0.01
Random State	42

Model Hyperparameters



To control the training iterations to avoid underfitting and overfitting to influence model convergence.

To adjust the step sizes to ensure efficient model learning and stable performance as it impact the gradient descent convergence.



To explores various combination of hyperparameters values within a predetermined grids

```
The best overall R2 score is: 0.7443
The optimal hyperparameters is: {'learning_rate': 0.01, 'epochs': 300}
```

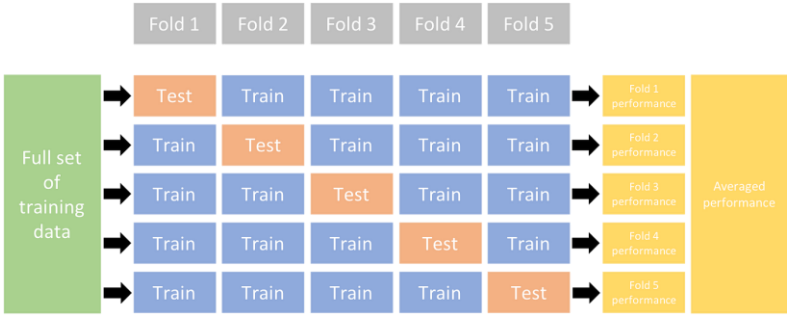
Best Result of Model Hyperparameters (Grid Search CV)

Evaluation Techniques

Evaluation Techniques

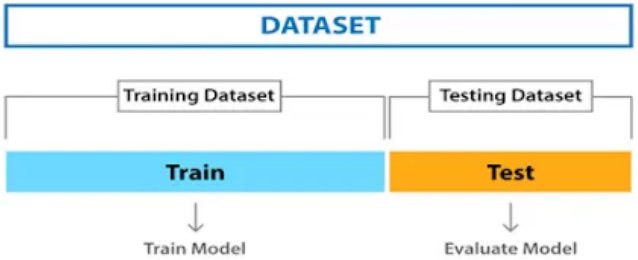
K-Fold Cross-Validation

Splits data into k subsets to train and validate models k times for accuracy



Holdout Validation

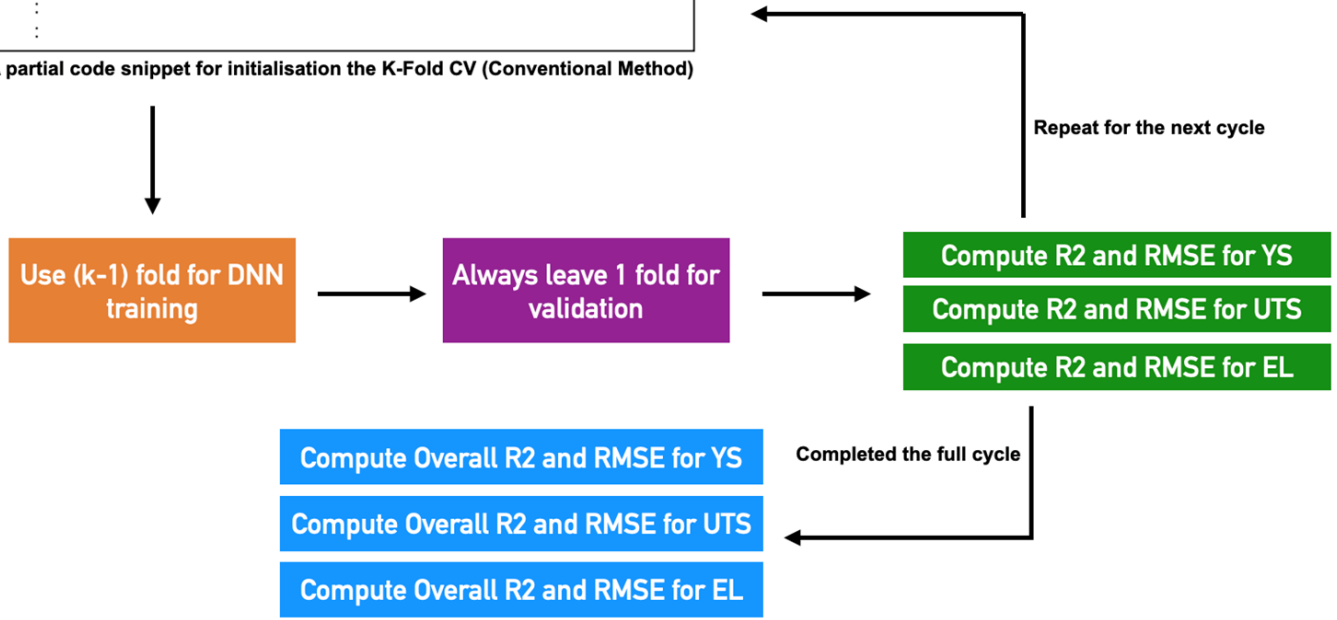
Splits data into training and testing sets



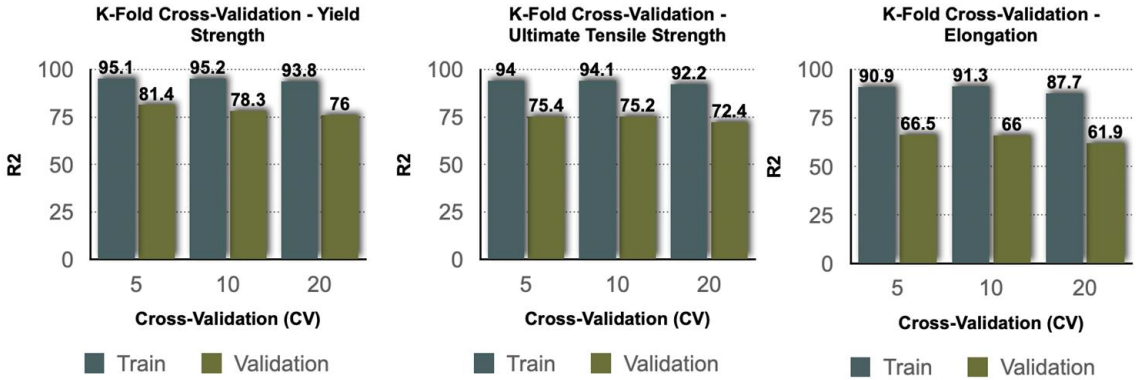
Evaluation Techniques (K-Fold CV)

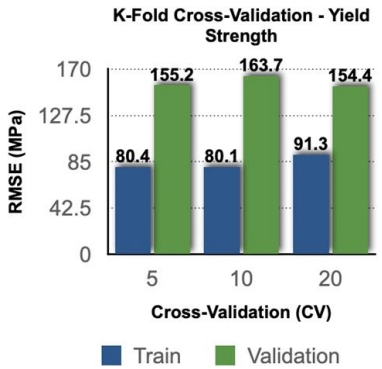
```
from sklearn.model_selection import KFold

cv = KFold(n_splits = 5, random_state = 42, shuffle = True)

for train_index, val_index in cv.split(X_train_ed):
        X_train, X_val = X_train_ed.iloc[train_index], X_train_ed.iloc[val_index]
        Y_train, y_val = y_train_ed.iloc[train_index], y_train_ed.iloc[train_index]
        :
        :
```

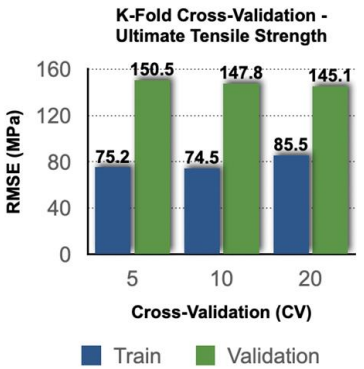
A partial code snippet for initialisation the K-Fold CV (Conventional Method)

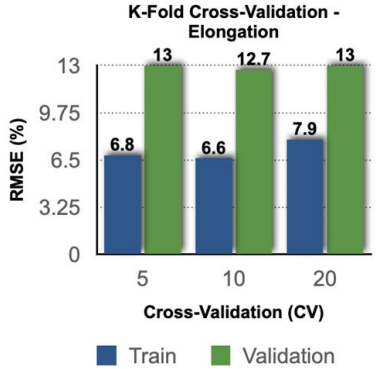


Evaluation Techniques (K-Fold CV Results)









Findings:

- **Bias-Variance Trade-Off:** More folds increase variance and reduce generalization.
- **Optimal K-Fold:** 5-fold balances training and validation performance.

Recommendations to Reduce Overfitting:

- Remove redundant and irrelevant features from the model
- Add more diverse set of new data
- Sampling Techniques

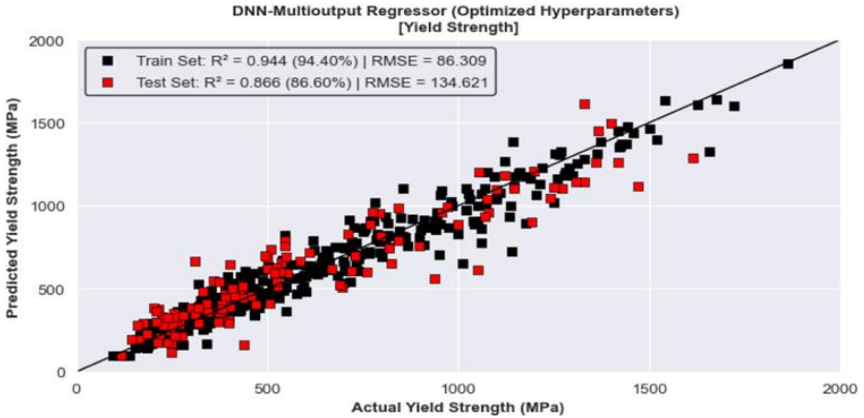
Evaluation Techniques (Holdout Validation)

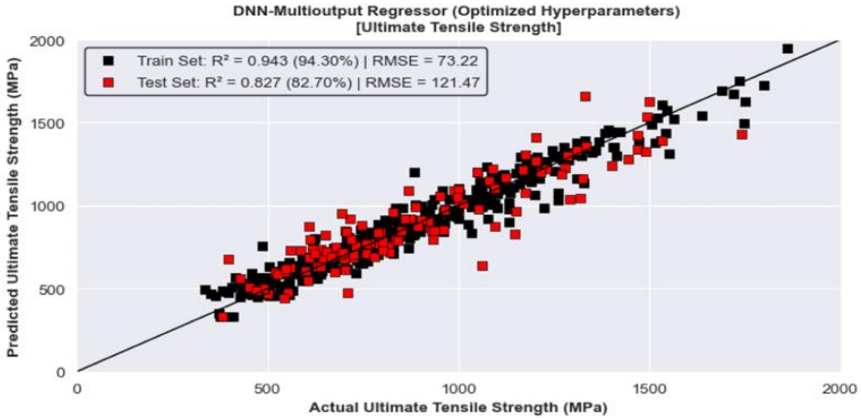
```
# training set for training
with tf.device("/CPU"):
    # define base model
    dnn_gs_ed = KerasRegressor(model = dnn_gs_mo,
                               loss = custom_mse_loss,
                               batch_size = 32,
                               epochs = 300,
                               optimizer = Adamax(learning_rate = 0.01),
                               metrics = [R2Score, RootMeanSquaredError],
                               random_state = 42)
    # fit the DNN regressor to the training data
    dnn_gs_ed.fit(X_train_ed, y_train_ed, shuffle = True)
```

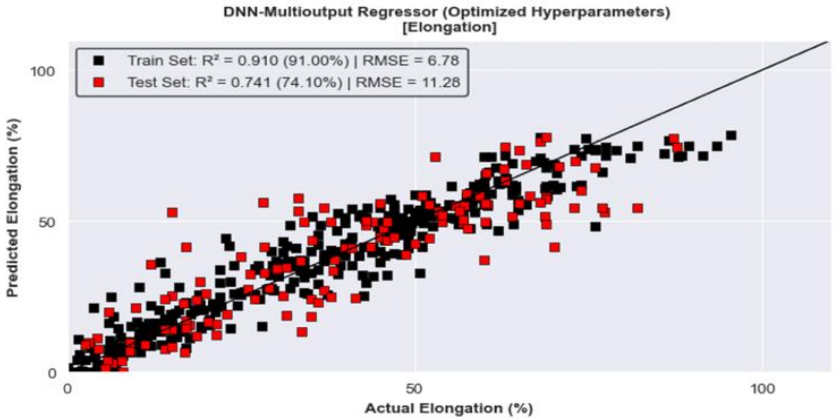
A code snippet of multi output regression training process

- Train DNN-multioutput regressor on 75% training data.
- Evaluate DNN-multioutput regressor on 25% testing set - R2 and RMSE.

Evaluation Techniques (Holdout Validation Results)

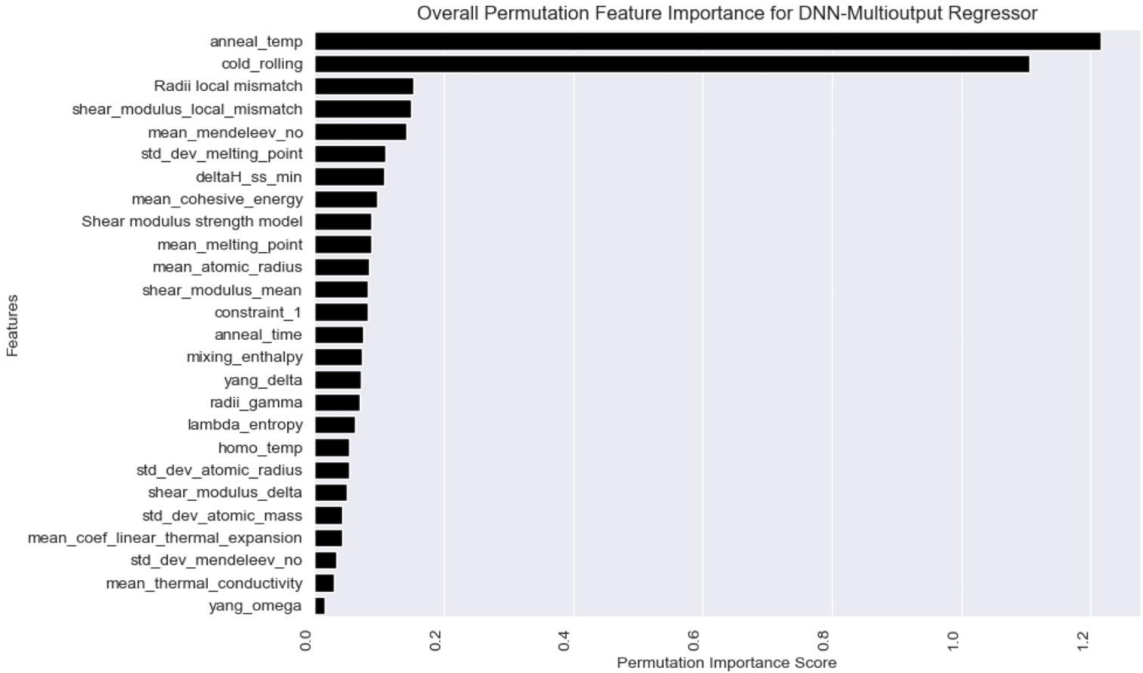


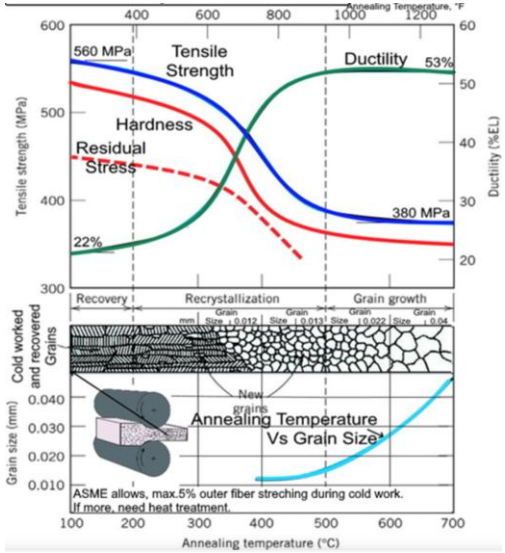




Permutation Feature Importances

- Randomly shuffle feature values one column at a time.
- Measure model performance before and after shuffling using MSE.
- Calculate the change in performance.
- Important features show higher performance change scores.



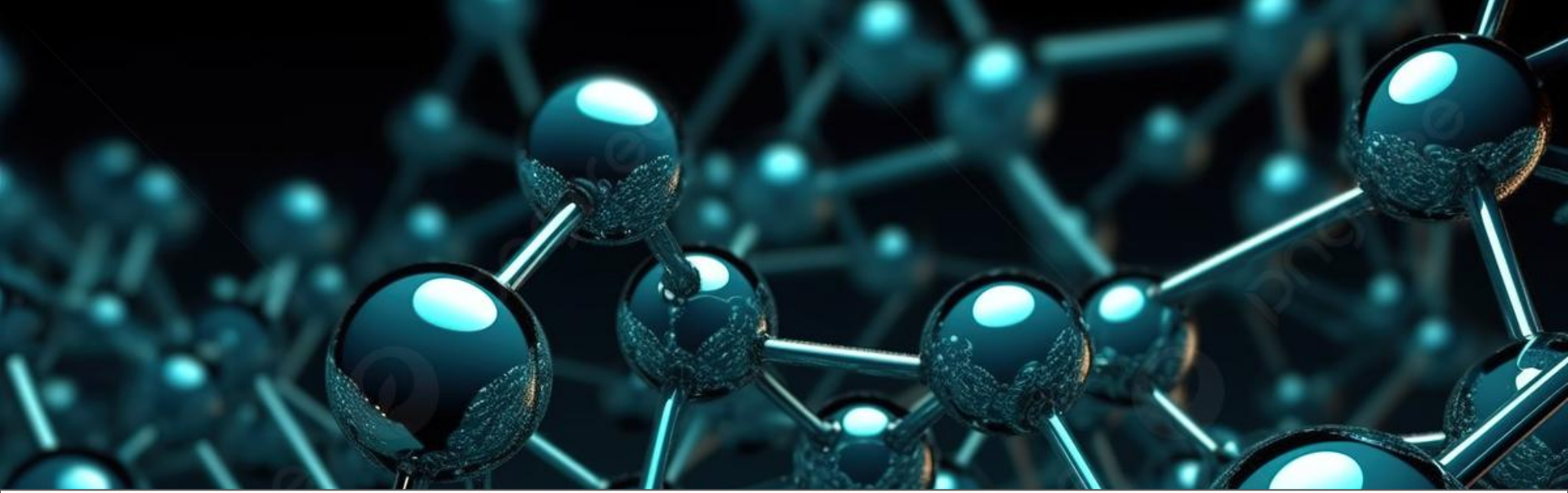


Cold rolling:

- Increases dislocation density, which enhances YS and UTS; May reduce elongation due to strain hardening effects.

Annealing Temperature:

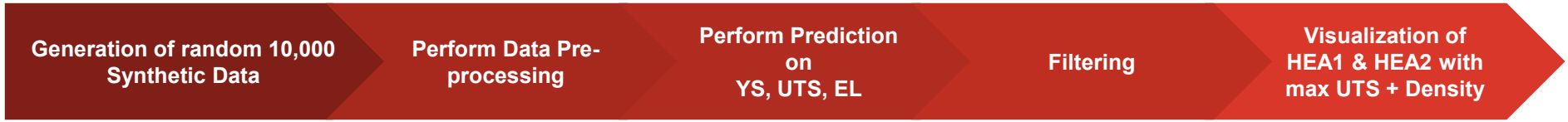
- Higher annealing temperature can lead to grain growth and reduces yield strength.
- Affects grain boundary strengthening mechanisms which impacts UTS.
- Higher annealing temperature can promote grain boundary sliding, which improves elongation.



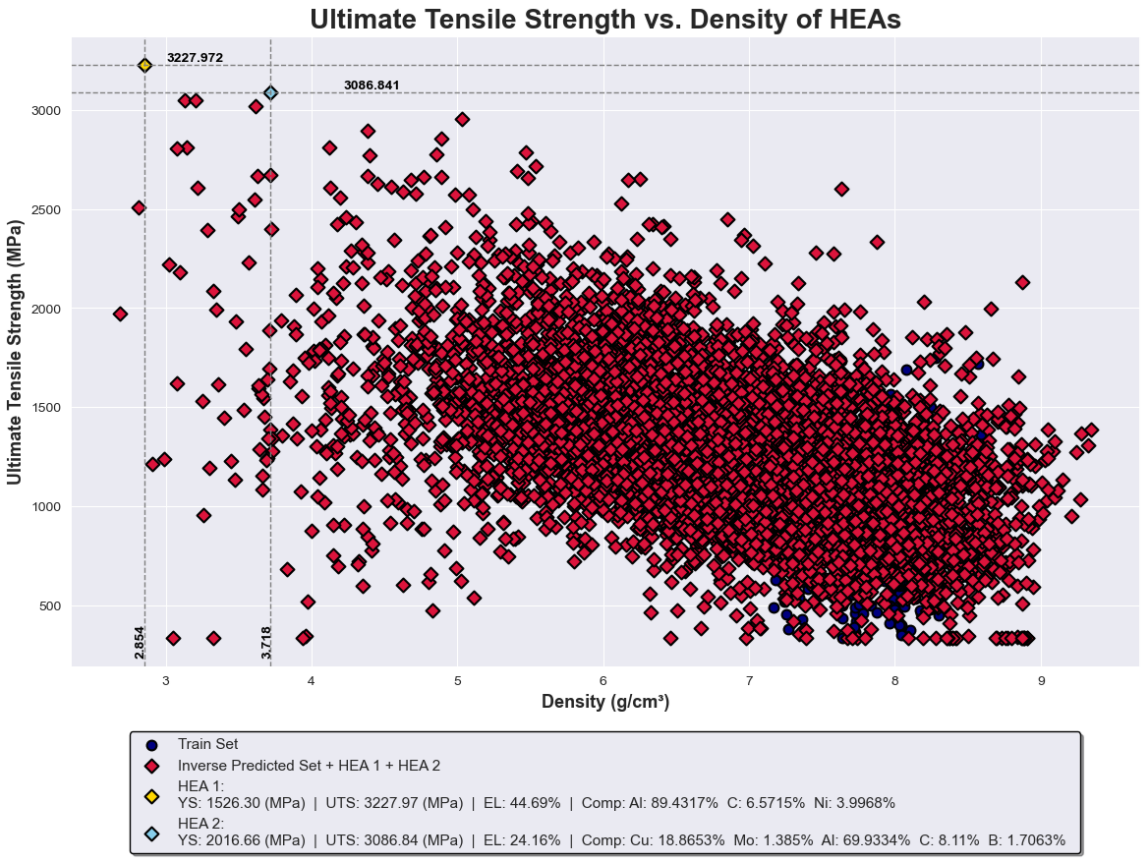
Segment 3: Inverse Prediction

Objective: Using the present work to design HEAs with high UTS and lighter density

Process of Inverse Prediction



Visualization of HEA1 & HEA2

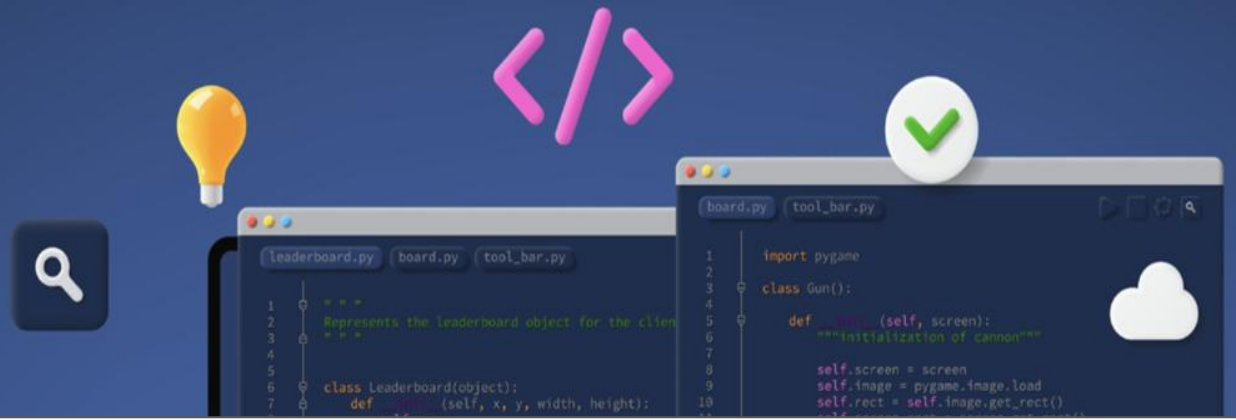


HEA 1

	4404
Co	0.000000
Ti	0.000000
V	0.000000
Cu	0.000000
Mo	0.000000
Al	89.431700
Cr	0.000000
C	6.571500
Mn	0.000000
B	0.000000
Fe	0.000000
Ni	3.996800
cold_rolling	40.000000
homo_temp	1373.000000
anneal_time	4.000000
anneal_temp	673.000000
constraint_1	1492.000000
Density	2.853741
pred_YS	1526.295800
pred_UTS	3227.972000
pred_EL	44.689667

HEA 2

	5804
Co	0.000000
Ti	0.000000
V	0.000000
Cu	18.865300
Mo	1.385000
Al	69.933400
Cr	0.000000
C	8.110000
Mn	0.000000
B	1.706300
Fe	0.000000
Ni	0.000000
cold_rolling	93.000000
homo_temp	1493.000000
anneal_time	0.125000
anneal_temp	773.000000
constraint_1	59.125000
Density	3.717963
pred_YS	2016.656000
pred_UTS	3086.841300
pred_EL	24.159077



Segment 4: Web Application using Streamlit

Objective: To allow research scientists to interact with DNN-Multi output Regression for informed decision-making and problem-solving

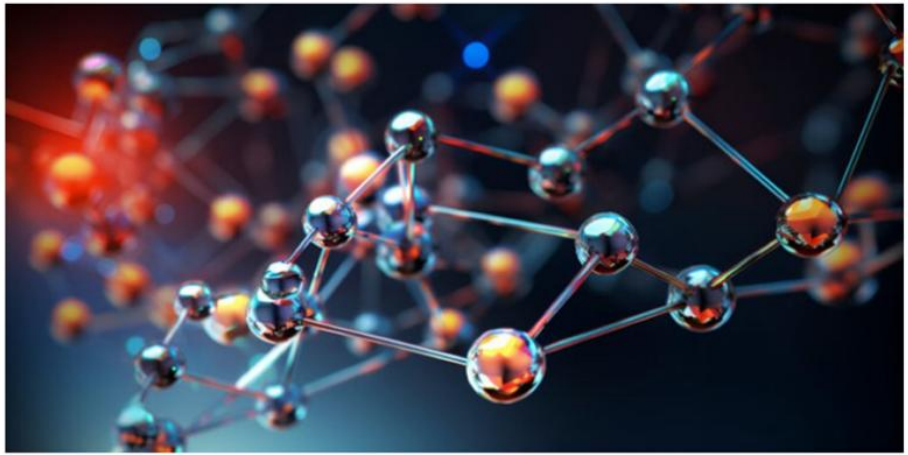
localhost:8501

CAP Calculator | STEP | CourseRekt | NTUMods - Ha... | Post | LinkedIn | Time Series Analy... | Time Series Forec... | amanpreetsingh45... | All Bookmarks

Deploy

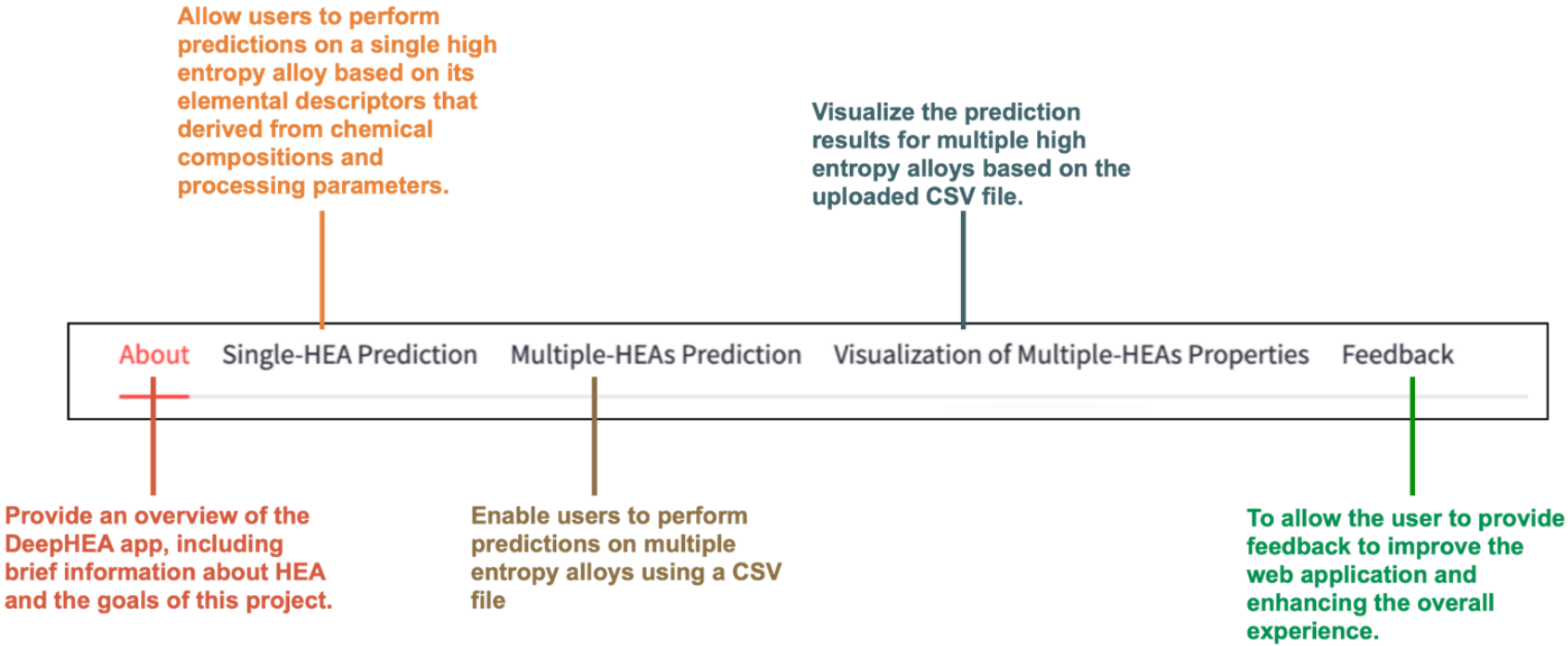
DEEPHEA

Unlocking Materials Potential: Predicting HEA Properties with Advanced DNN Solutions



About | Single-HEA Prediction | Multiple-HEAs Prediction | Visualization of Multiple-HEAs Properties | Feedback

Functionalities



Tutorials and Documentation