Week 11: A case study

M P Gururajan, Hina Gokhale and N N Viswanathan

Department of Metallurgical Engineering & Materials Science
Indian Institute of Technology Bombay

October, 2024

Machine learning based prediction of metal hydrides for hydrogen storage, part I: Prediction of hydrogen weight percent

A Rahnama, G Zepon, and, S Sridhar

International journal of hydrogen energy, **44**, 7337-7344, 2019.

The problem

- Hydrogen storage: how to?
- *The alloying strategy would be to optimise the H/M (Hydrogen/ Metal ratio in the chemical formula for the hydride) ratio to be as high as possible, have an efficient charging and dis-charging capacity and is able to operate as close to ambient temperature and pressure. In addition, cost and raw material availability would become a factor for deployment.*
- Hydrogen storage capacity: target!

Issue

- Reproduce the figures / tables
- Data used in the study: US DoE – Hydrogen Storage Materials Database
- The database: not accessible
- Very similar but not the same data
- Data and code curation: how to?

Figure 1a: from paper

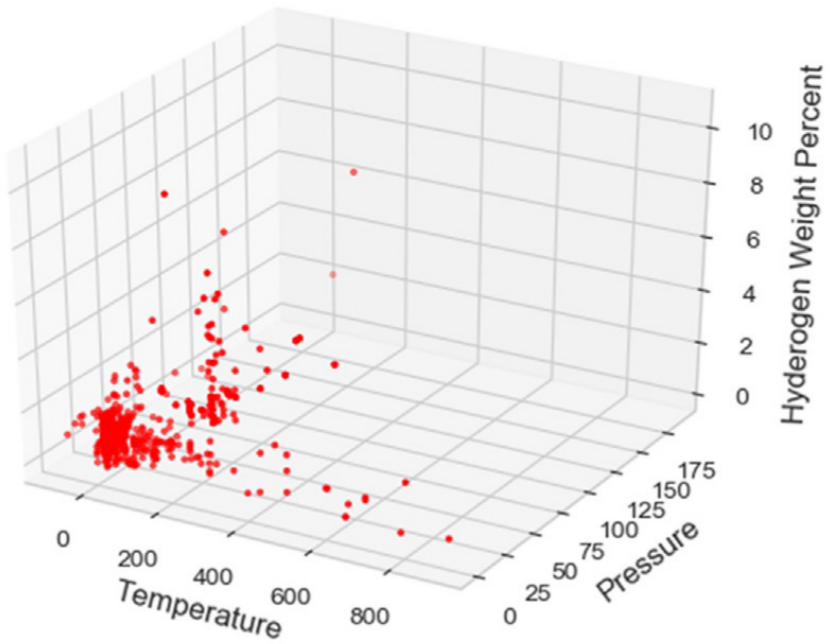

Figure: Figure 1a from the paper – a 3-D plot.

3-D plotting

```
import pandas as pd
import matplotlib.pyplot as plt
Data = pd.read_csv("HydDataML.csv")
fig = plt.figure()
ax = fig.add_subplot(projection="3d")
ax.scatter(Data["Temperature_oC"],
Data["Pressure_Atmospheres_Absolute"],
Data["Hydrogen_Weight_Percent"], marker="s",color="r")
ax.set_xlabel("Temperature (in deg C)")
ax.set_ylabel("Pressure (atmos)")
ax.set_zlabel("Hydrogen weight percent")
plt.show()
```

Figure 1a: reproduction

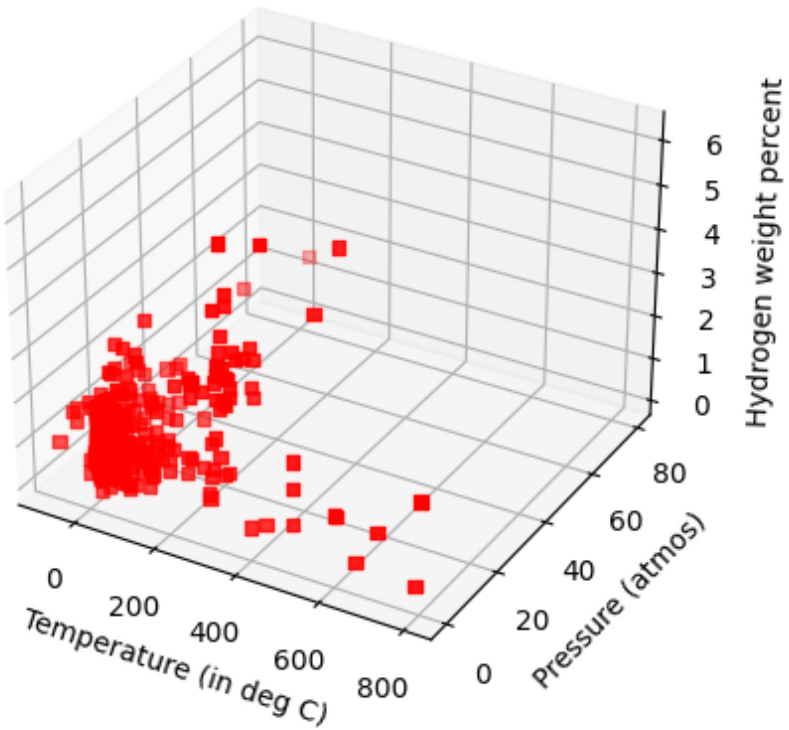

Figure: Reproduction of Figure 1a – note that when plotted, the figure can be rotated at the matplotlib interface.

Figure 1b: from paper

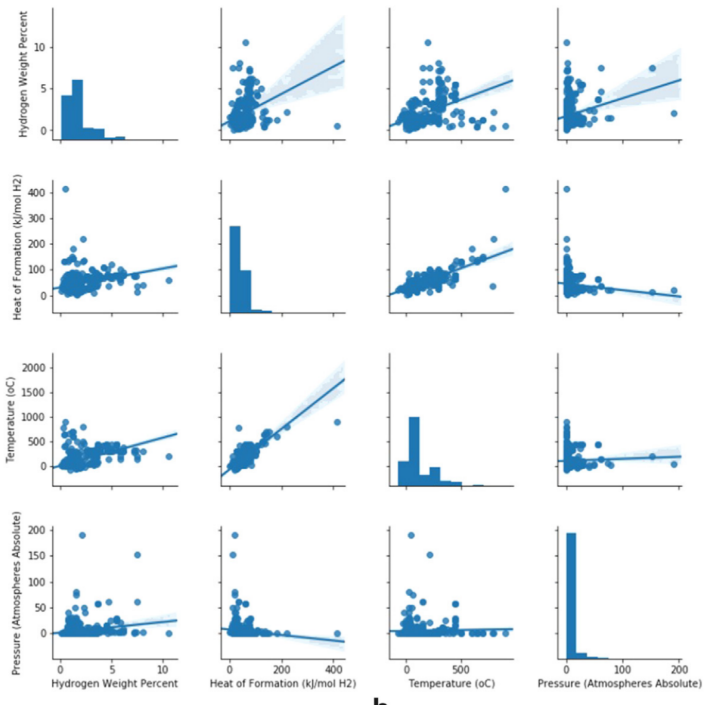

Figure: Figure 1b from the paper – a pair plot of the relevant variables.

Pair plot

```
import seaborn as sns
Data=data[["Hydrogen_Weight_Percent",
"Heat_of_Formation_kJperMolH2",
"Temperature_oC","Pressure_Atmospheres_Absolute"]]
sns.pairplot(Data)
plt.show()
```

Figure 1b: reproduction

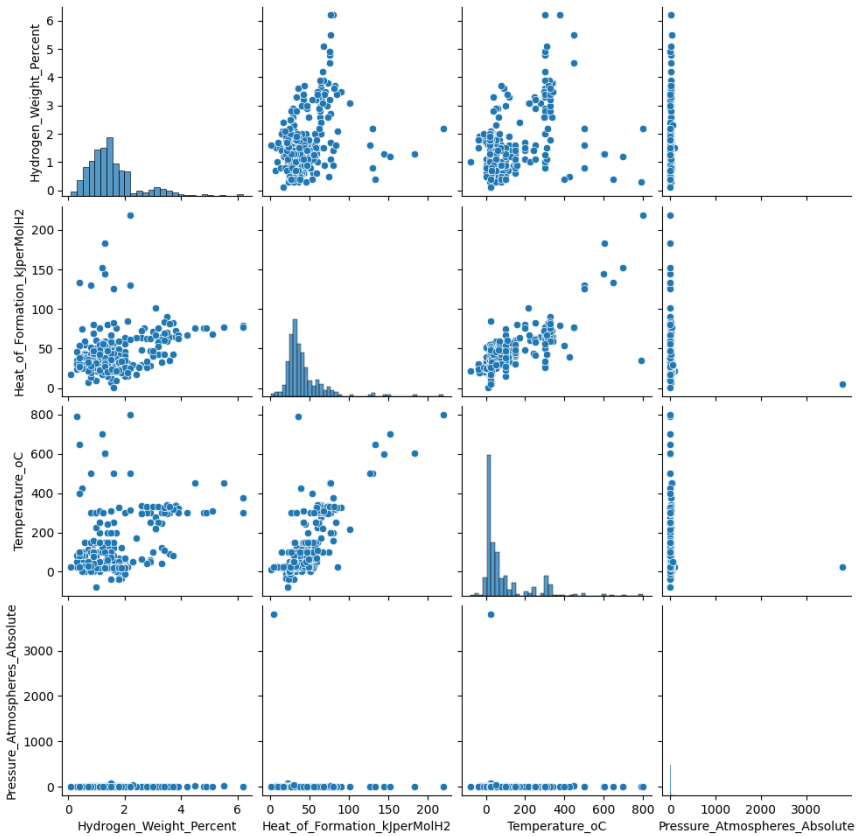

Figure: Reproduction of Figure 1b – note that the differences you see in the pressure plots are due to an outlier.

Outlier removal from data

From the paper:

> *Outlier are important to identify because they affect the bias (intercept) of the regression models. In the current analysis we classified the outlying points as being located within a normal distribution of residuals or not. If points were found outside the normal distribution of residuals they were discarded.*

Figure 2: from paper

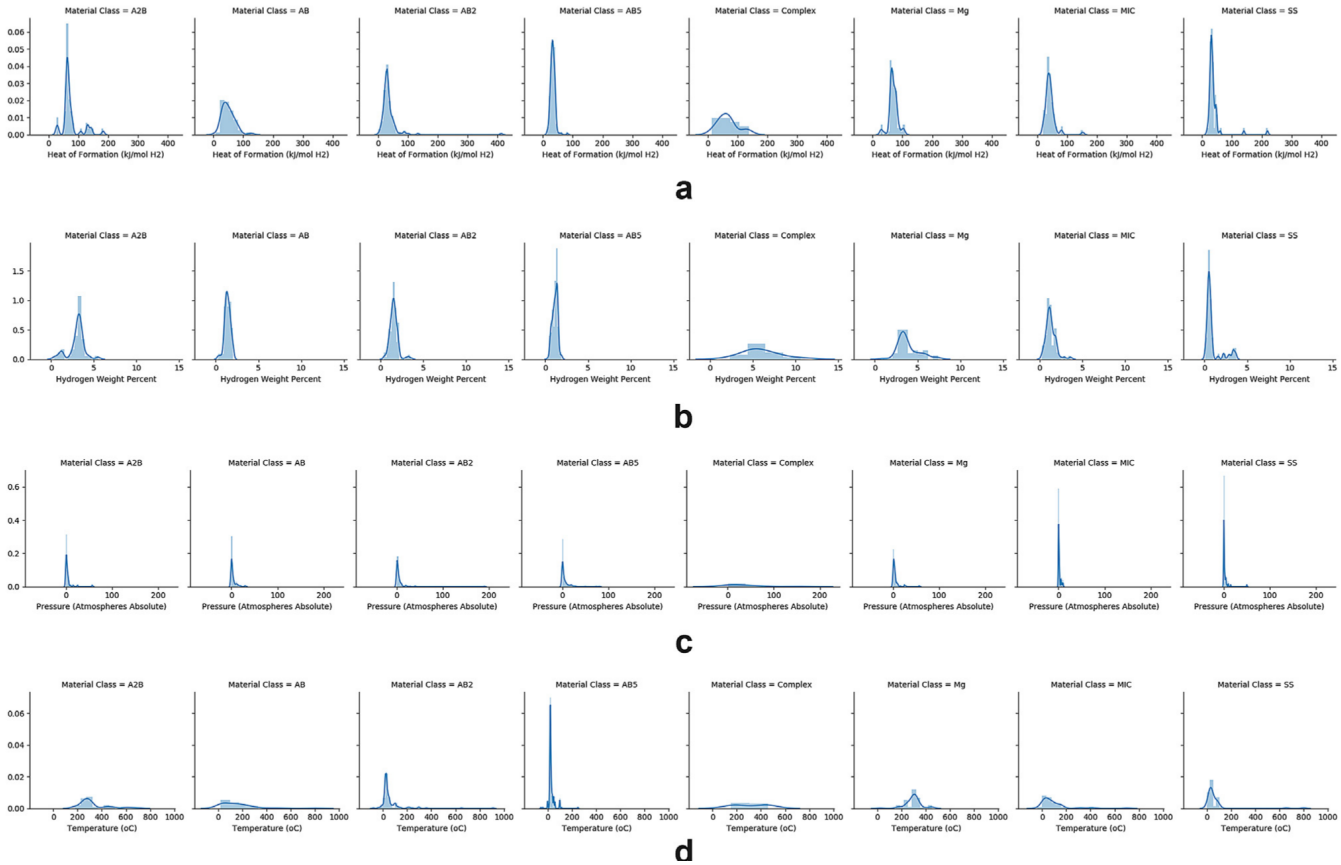

Figure: Figure 2 from the paper – plots according to material class of the variables.

Plotting by material class

```
fig, ax = plt.subplots(4,7,layout="constrained")
i = 0
for category, data in data.groupby('Material_Class'):
    sbn.histplot(data['Heat_of_Formation_kJperMolH2'],
kde=True,ax=ax[0,i])
    ax[0,i].set_title(f'Material Class: {category}')
    ax[0,i].set_xlabel('Heat of formation')
    sbn.histplot(data['Hydrogen_Weight_Percent'],
    kde=True,ax=ax[1,i])
    ax[1,i].set_xlabel('Hydrogen wt pt')
    sbn.histplot(data['Pressure_Atmospheres_Absolute'],
    kde=True,ax=ax[2,i])
    ax[2,i].set_xlabel('Pressure (atmos)')
    sbn.histplot(data['Temperature_oC'],kde=True,ax=ax[3,i])
    ax[3,i].set_xlabel('Temperature (deg C)')
    i = i + 1
plt.show()
```

Figure 2: reproduction

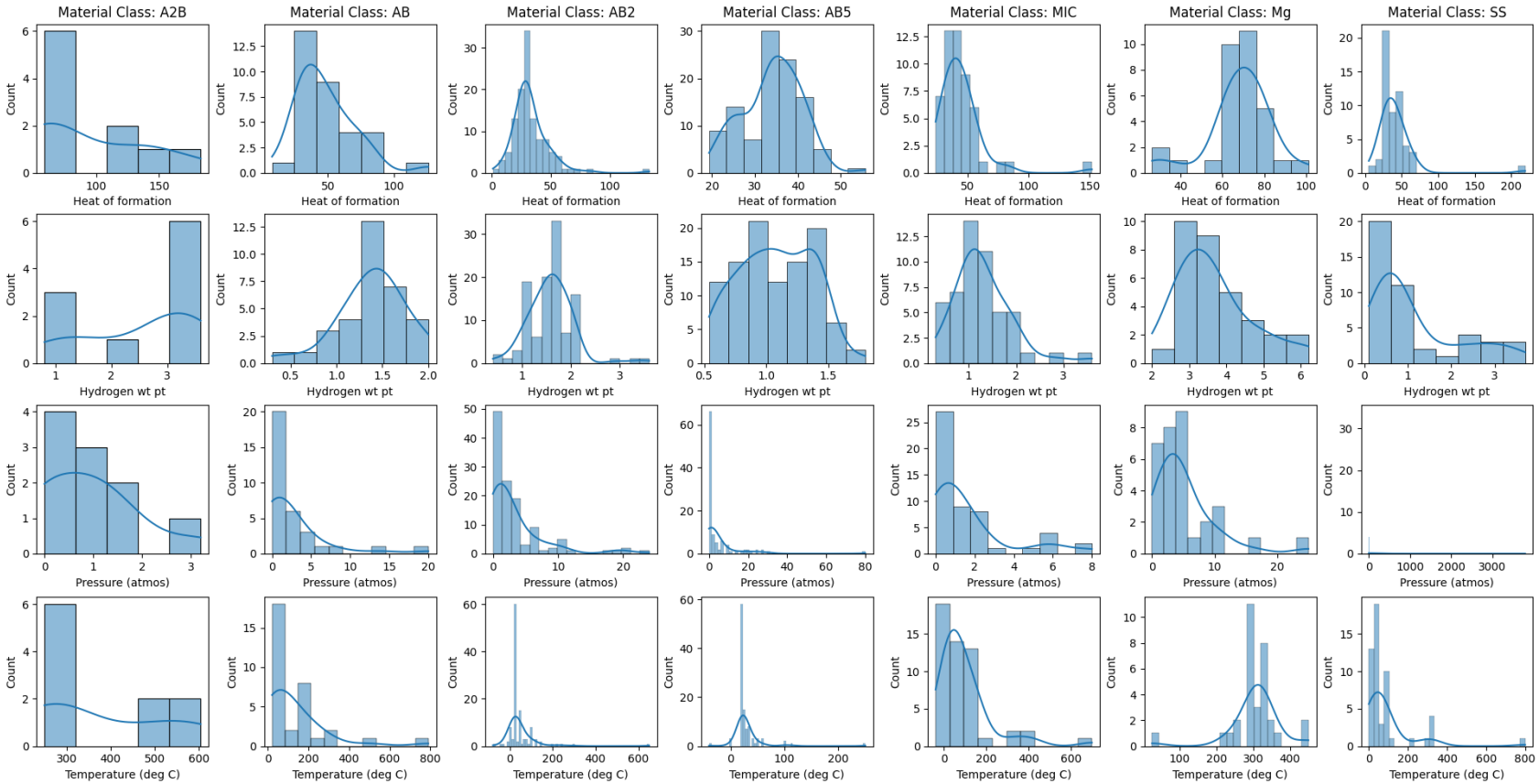

Figure: Reproduction of Figure 2 – note that the complex class is missing in our data.

Dimension reduction

- Recall PCA: can reduce dimensions
- Not interpretable
- Is there an interpretable way of deciding on the importance of different features?
- Permutation feature importance
- *For determining the feature importance we used permutation feature importance for which we had an out-of-bag dataset and we randomly permuted the value of the feature j and computed the permuted error (error -j;permuted) and compared it with overall error (error - overall). The features were, then, ranked according to the increase in the overall error when they were permuted.*
- Bagging: sub-sampling with replacement

Feature importance

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn.ensemble import RandomForestClassifier
from sklearn.inspection import permutation_importance

data = pd.read_csv('HydDataML.csv')
#print(data)
Data = data[['Material_Class','Composition_Formula',
'Hydrogen_Weight_Percent','Heat_of_Formation_kJperMolH2',
'Temperature_oC','Pressure_Atmospheres_Absolute']]
#print(Data)
ClnData = Data.dropna()
#print(ClnData)
```

Feature importance

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

EncodedClass =
encoder.fit_transform(ClnData['Material_Class'])
EncodedComp =
encoder.fit_transform(ClnData['Composition_Formula'])
#print(EncodedClass)
#print(EncodedComp)

EncFtr = pd.DataFrame({'Material_Class':EncodedClass,
'Composition_Formula':EncodedComp})
#print(EncFtr)
```

Feature importance

```
UnEncFtr = pd.DataFrame({'Heat_of_Formation_kJperMolH2':
 ClnData['Heat_of_Formation_kJperMolH2'],'Temperature_oC':
 ClnData['Temperature_oC'],'Pressure_Atmospheres_Absolute':
 ClnData['Pressure_Atmospheres_Absolute']})
#print(UnEncFtr)
Features = UnEncFtr.join(EncFtr)
#print(Features)

Target = ClnData['Hydrogen_Weight_Percent']

from sklearn.preprocessing import MinMaxScaler
scaling = MinMaxScaler()
scaling.fit(Features)
ScaledFeatures = scaling.transform(Features)
```

Feature importance

```
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.inspection import permutation_importance
from sklearn.model_selection import train_test_split

X = ScaledFeatures
y = Target
#X = Features
#y = Target

# 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)

# Train a Random Forest model
est = HistGradientBoostingRegressor()
est.fit(X_train, y_train)
```

Feature importance

```
result = permutation_importance(est, X_test, y_test,
n_repeats=10, random_state=42)

perm_importances = result.importances_mean
perm_std = result.importances_std
sorted_idx = perm_importances.argsort()
feature_names = Features.columns

Result = pd.DataFrame({'Importance': perm_importances,
'Std': perm_std},
index=feature_names[sorted_idx]).sort_values('Importance',
ascending=True)

print(Result)
```

Table 2: from paper and our analysis

Table 2 – Feature importance for predicting hydrogen weight percent.

Feature	Materials class	Temperature	Heat of formation	Pressure	Composition formula
Score	0.487527	0.323918	0.15273	0.100549	0.004816

```
(base) guru@BhaskarAngiras:~/.../Week11$ python3 Table2.py
                                Importance       Std
Composition_Formula               0.016396  0.009747
Heat_of_Formation_kJperMolH2      0.087576  0.041414
Pressure_Atmospheres_Absolute     0.088917  0.030915
Material_Class                    0.470032  0.111446
Temperature_oC                    0.899475  0.171196
```

Figure: Table 2 from the paper and our script – feature importance. From this, it is concluded that the composition is not a key variable for hydrogen weight percent.

Figure 3a: from paper

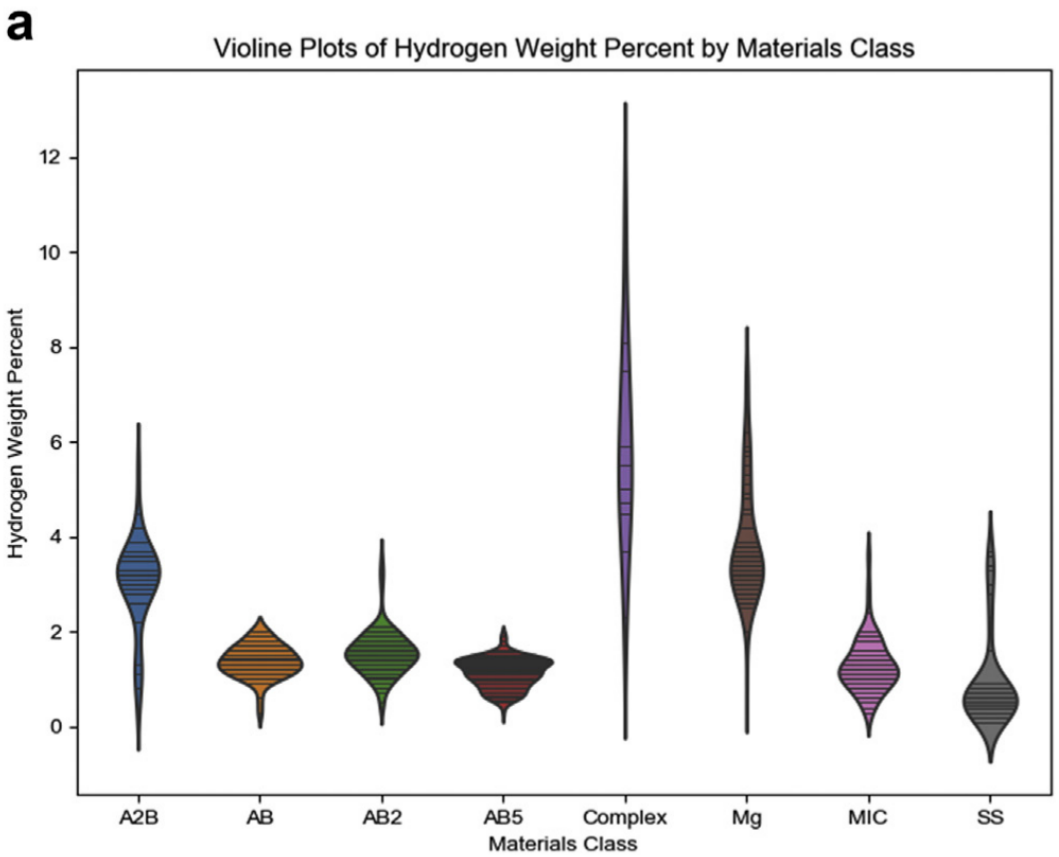

Figure: Figure 3 a from the paper – violin plots according to material class of the variables.

Violin plots

```
import seaborn as sns

sns.violinplot(data,x="Material_Class",
y="Hydrogen_Weight_Percent")
plt.xlabel("Material Class")
plt.ylabel("Hydrogen weight percent")
plt.show()
```

Note: Violin plots are box-plots with density plot on either side!

Figure 3a: reproduction

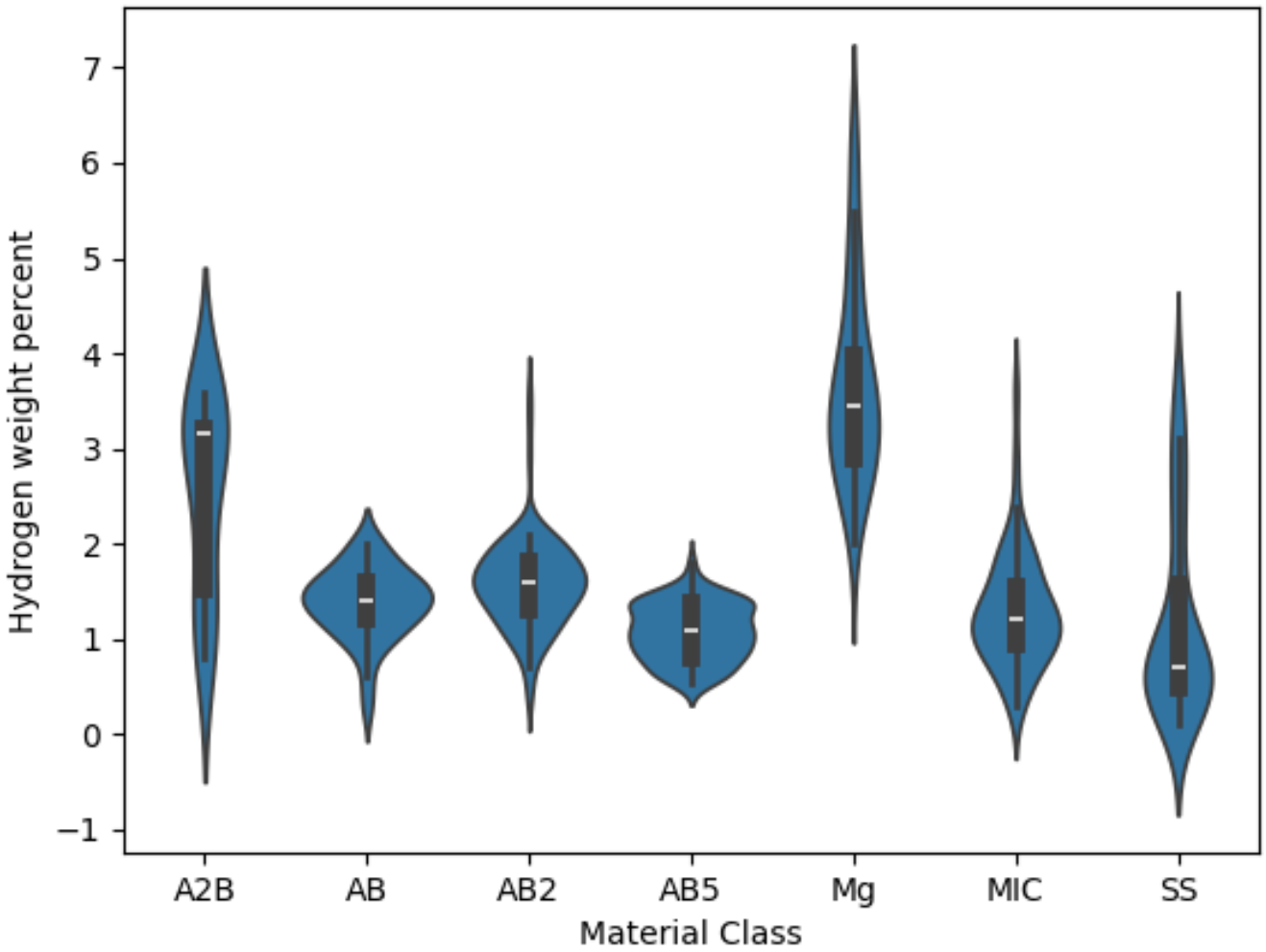

Figure: Reproduction of Figure 3 a – note that the complex class is missing in our data.

Figure 3b: from paper

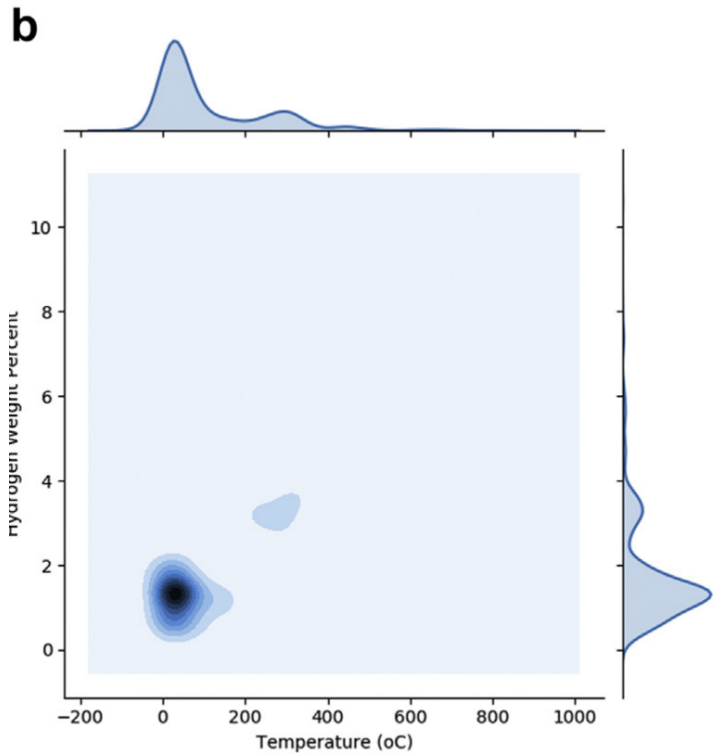

Figure: Figure 3 b from the paper – bivariate kernel density plots showing higher hydrogen weight percent comes at the cost of higher temperatures.

Bivariate kernel density plot

```
import seaborn as sns
sns.kdeplot(data,x="Temperature_oC",
y="Hydrogen_Weight_Percent",fill=True)
plt.show()
```

Figure 3b: reproduction

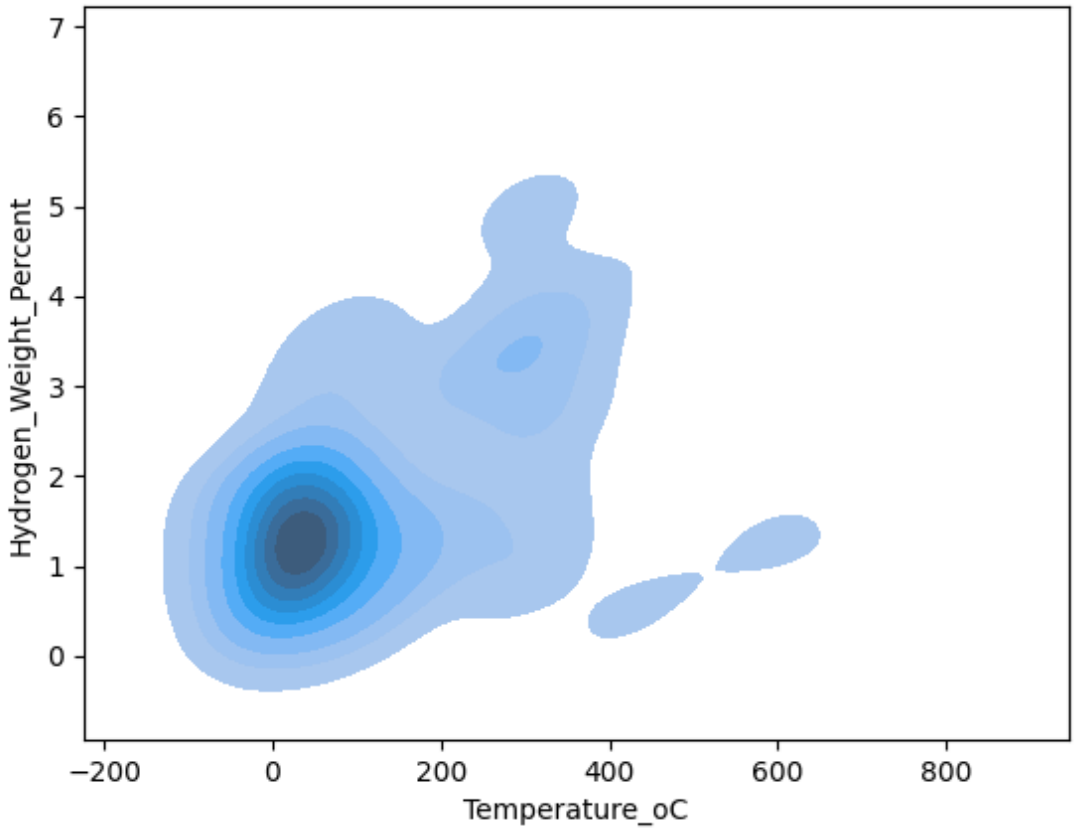

Figure: Reproduction of Figure 3 b – I could not find a way to plot the density on the axes.

Regression

- Linear regression
- Neural network regression
- Bayeisan linear regression
- Boosted decision tree regression

Boosting

From wikipedia page – with reference to image classification:

1. Form a large set of simple features
2. Initialize weights for training images
3. For T rounds
    1. Normalize the weights
    2. For available features from the set, train a classifier using a single feature and evaluate the training error
    3. Choose the classifier with the lowest error
    4. Update the weights of the training images: increase if classified wrongly by this classifier, decrease if correctly
4. Form the final strong classifier as the linear combination of the T classifiers (coefficient larger if training error is small)

Can modify for any other task!

Summary

- EDA: a very important part of dealing with data
- Smaller data sets and subject matter expertise: interpretable ML is key
- Python learnt so far: with some effort you can largely understand and reproduce results from literature – albeit 5 years old!
- Data collection and curation / reproducible programming: important

Thank You!

Questions, clarifications, comments?