CHEM452/CHBE413

Chemical Data Science and Engineering

Homework 3

Due Date: September 18th, 2025

Please complete and submit this homework as a Jupyter notebook. *For every plot your axes must be labeled, and if there are multiple data sets on the same plot, a legend must be provided!* *Please split your code into several blocks (at least one block for each sub-question, avoiding crowding your code into a single chunk) and offer appropriate print for your double-check and also for my review.*

1. **A Molecule is a Bag of Elements!** (dataset = *Tm\_200\_subset\_original.csv*).
2. (**4 points)** Next week, we will discuss a variety of approaches to represent chemical information in machine learning models. Here, you will create a basic representation that converts a molecule, reported as a SMILES string, into a numerical vector that we can work with. You will use a “Bag-of-Elements” vector that can be constructed for each SMILES string (this is a variation on the extremely common “Bag-of-Words” approach <https://en.wikipedia.org/wiki/Bag-of-words_model>) used in text analysis. Excluding hydrogen, which you will not worry about, there are ten unique elements across all SMILES strings in the dataset:

[‘C’, ‘O’, ‘N’, ‘Cl’, ‘B’, ‘Br’, ‘I’, ‘F’, ‘S’, ‘P’]

Write a function in Python that converts the SMILES string to a numerical “Bag-of-Elements” vector. The vector will be a tuple of 10 natural numbers (positive integers and 0). Each component in the vector will report the number of times that the element appears in the SMILES string of a molecule. For a SMILES string given by “CCCCC”, the vector would have a 5 in whatever component corresponds to “C” and 0 everywhere else. Note that you will have to be careful with how you handle Br vs B and Cl vs C.

*Hint: Have your function take as input a SMILES string that you would like converted to a bag-of-elements vector. Within the function, start by defining a list named search\_elements that contains all of the elements you are looking for, and make sure that it is ordered so that the list has the longest elements (2 letters) at the beginning. Also, define a bag-of-elements vector as a numpy vector of zeroes. Then, for a given SMILES string that your function takes in, iterate through search\_elements, checking the SMILES string for the presence of each element. If you find that element in the SMILES string, add 1 to the element of your bag-of-elements vector corresponding to that character, and then delete or replace the character in the SMILES string. Continue iterating until you have made it through your list of search\_elements. At the end, return the bag-of-elements vector.*

1. (**4 points)** Write a function to compute the norm between two Bag-of-Elements vectors using the dot product as a metric for the similarity/dissimilarity between two molecules. *Note: the Bag-of-Elements vector is not normalized - try normalizing the vectors and checking if that changes your results.* Identify the three “most similar” and the three “most dissimilar” pairs of molecules in the dataset based on this distance metric and plot the 2D molecular structures in your Jupyter notebook (see RDkit instructions in the last homework). What limitations might there be to using this similarity metric? What features of molecules are not included in the Bag-of-Elements approach?
2. (**2 points)** Repeat (b) for a different metric: the L2 norm of the difference between two vectors. How does changing the concept of distance impact your findings?
3. **Linear Regression for Melting Points.** (dataset = *Tm\_200\_subset\_original.csv*).

Now we will test how successfully we can predict the Tm of a molecule based on its chemical structure using the “Bag-of-Elements” molecular representation.

1. (**3 points)** First, we must guard against bias by dividing up the 201 training points into a training and a test set to reduce model overfitting. Split the original 201-point dataset into two datasets, one that is 80% and called the “training” data and the other which is 20% called the “test” data. Make the split with the first 160 data points in the file being for training, and the next 41 datapoints being for testing. Make sure that each molecule only appears in **either** the training or test data sets, and not both. Report the mean and standard deviation of Tm, the number of atoms, dipole moment, and quadrupole moment in both the training and test sets. Is the data in your test set representative of that in your training set?
2. (**4 points**) Using the scikit-learn implementation (*sklearn.linear\_model.LinearRegression*), fit three linear regression models to the training datasets and then assess their performance on the test data sets: one that correlates Tm vs the number of atoms, one that correlates Tm vs dipole moment, and one that correlates Tm vs quadrupole moment. Report the relative goodness of fits using the coefficient of determination (R2) and the root mean squared error (RMSE) for the training and test datasets for each of the three models. Use sklearn’s metrics library (<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>) to do this. Does dipole moment, quadrupole moment, or number of atoms correlate most strongly with Tm? In each model is the performance on the training or test data superior?
3. (**3 points**) Last, we will extend our method to multilinear regression by using our “Bag-of-Elements” vector as the feature for prediction. Use the scikit-learn linear regression routine in (b) to fit a model on the training set using the Bag-of-Elements vector, and then apply this to the held-out test set. Report the RMSE and R2 on the training and test sets.

*Hint: If you are having trouble figuring out how to do multilinear regression, try consulting the scikit-learn documentation and looking through some of their examples! This is a good life skill to possess.*