**Seed Data Formatting Changes**

LCCCSeed-ML-2-2-25-RemovedCommasFromSolvents.xlsx  
Starting with the above file, there are three changes between this version and the One drive version:

1. is that “Solvents” column in the OneDrive version is separated by commas while I needed to switch to a semicolon separation due to some solvents having commas in their names.
2. One drive version contains an invisible character in one of the “Injected ….” columns – (I can’t remember which, and I fixed using excel so no clear log)
3. I standardized the Solvents columns in the local version so that the highest ratio solvent was always listed first, both in the “Solvents” column, but also in the “Solvent Ratio” to match the ordering. Also, there was one entry whose ratio was 1:1 which was changed to 50, 50.

If these changes will not affect your website related scripts, maybe I could recheck no unwanted changes between files and replace the LCCCSeedDataMain-toML with my version.

**RDkit Descriptors Generation**

**Script:** RDkit-DescriptorGenerator.py

**Input files:** LCCCSeed-ML-2-2-25-RemovedCommasFromSolvents.xlsx

**Output file:** SolventDescriptors-UPDATED-2-23-25.xlsx

This script takes my version of the Seed Data – parses the solvents from the Solvent column. Then uses PubChempy package to find the SMILEs for the unique solvents. After which each SMILEs are ran through RDkit package to generate both the molecular descriptors and the morgan fingerprint descriptors. Note that only 196 molecular descriptors of the 212 molecular descriptors available were generated – this refers to polymer and solvent descriptors. I am only getting 118 solvent descriptors. This was done to mimic Ethier’s paper usage – they cut out repetitive descriptors.

**Updated Feb-23:** I had accidently left out one molecular descriptor “NumAromaticRings” after double checking Ethier2022 paper descriptors.

**Preprocessing Outline**

**Script:** PolystyrenePreprocessing.py

**Input files:** 1. LCCCSeed-ML-2-2-25-RemovedCommasFromSolvents.xlsx

2. SolventDescriptors-UPDATED-2-23-25.xlsx

**Output file:** polystyrene-imputated-molecular-descriptor-includes-inj-volume-2-15-25.xlsx

The goal of this script is:

1. Check how percentage of missing entries in each column and drop columns that have more than 50% missing.
2. Remove all the columns related to quality scoring – the quality score itself is kept - and remove paper writing notes.
3. Sort out only polystyrene entries to be used.
4. Solvent-RDkit-Descriptors-2-2-25.xlsx   
   does not contain all the solvents in our database, only 50 solvents currently. There are a few trickier ones that I haven’t been able to figure out the SMILES for confidently – eg. Deuterated acetone. But, the majority of the solvents are listed, followed by their SMILEs which were used by the RDkit script to generate both the molecular descriptors and the morgan fingerprints. The file consists of ~1200 columns, the first 118 after the Smiles column are molecular descriptors followed by ~ 1000 morgan fingerprints.

Only one type should be used at a time, or it would be repetitive in nature. Therefore, molecular descriptors were chosen due to the reduced dimensionality and the morgan fingerprints were dropped. \*I do have a separate version that uses Morgan fingerprints and drops molecular descripts.

1. The “Solvents” column from my version of the seed data was separated and parsed to split the solvents into individual columns, Solvents\_0, Solvents\_1, Solvents\_2, Solvents\_3. Each containing one of the solvents from the pairings “Solvents” column.
2. The data frame containing the parsed solvents needed to be cleaned down to only binary pairs to be used in the initial polystyrene ML model. Therefore, outlier cases such rows missing Solvents\_1 (due to Solvents\_0 being 100% of the ratio), and rows containing Solvents\_2, and Solvents\_3 were dropped.

**Preprocessing Outline Cont.**

1. The parsed binary solvents were also stripped of leading and trailing whitespaces. As well as “ (near crit)” which was dropped and not tracked.
2. New data frames were made to store the columns Solvents\_0 and Solvents\_1 separately. Indexes were reset. And then the data frame containing the molecular descriptors was merged on to the two new data frames, combined\_descriptors1 and combined\_descriptors2. This was done so that the Solvent names were matched, and descriptors were only appended if their associated solvent was matched.
3. The combined\_descriptors 1 and 2 had their indexes reset and their string columns dropped, “Solvents\_0” or “Solvents\_1” followed by “Solvents” and “SMILES” the RDkit file’s columns. The data frames containing only numerical data of molecular descriptors were then combined, row and column index to row and column index.
4. The dataframe consisting of ~200 columns of concatenated numeric molecular descriptors was then appended on to the earlier data frame that contained the binary solvents names parsed into separate columns.
5. Now, my version of the SeedData cleaned from 50% missing columns needed to undergo the same cleaning process regarding limiting it to only binary solvent combinations. Then the combined molecular descriptors data frame could be concatenated preserving the order.
6. I also wanted to retain as much experimental data as possible, so instead of dropping the detector column. I created a function to count the number of detectors to replace the string column.
7. Solvent ratio, particle diameter, pore size were all parsed from multiple entries in one column to separate columns.
8. The temperature column had its missing values filled with 25.

The “Pore size (°A)\_2” and “Particle diameter (μm)\_2” columns had their missing values filled with 0. Debating if 0 or a transient fill would be better, do you have any thoughts?

1. The categorical columns, "Stationary Phase", "Base Material", "Base Material Modification", "Phase" were all one hot encoded. I believe other categorical columns such as the manufacturer column were dropped during the 50% missing processing.

**Preprocessing Outline Cont.**

1. The data frame “Imputated” (already had temperature=25, and pore/particle filling) was ran through SciKit learn’s iterative imputer using XGboost as the regressor to fill the other missing experimental data. This was to preserve as much of the experimental columns as possible for ML usage. I do believe that XGboost may be overfitting here – but my thought is that it will be better or similar to mean filling imputation. I did try other regressors – BayesianRidge is the default, it did horribly.
2. polystyrene-imputated-molecular-descriptor-includes-inj-volume-2-15-25.xlsx file output from the preprocessing ready for ML usage.

**Preprocessing - Solvents Hot Encoded by Ratio and Data Imputation**

**Script:** Polystyrene-Preprocessing-MolDesc-Solvent-Hot-Encoding-2-21-25.py

**Input files:** 1. LCCCSeed-ML-2-2-25-RemovedCommasFromSolvents.xlsx

2. SolventDescriptors-UPDATED-2-23-25.xlsx

**Output files:** polystyrene-imputated-solvents-hot-encoded-2-23-25.xlsx

This is the upgraded version of preprocessing that hot encodes the solvents by their ratios before data imputation is done. This will be better scalable for the future; ideally allowing for ratio prediction to also predict which solvents to use. Processing is done to subset only polystyrene entries of binary solvent pairs.

**XGBoost Polystyrene Solvent Ratio Prediction**

**Script:** 1. XgBoost-Polystyrene-MolecularDesc-with-feature-importance.py

2. Unfinished-XgBoost-Nested-5-Fold-Validation-version.py

3. Unfinished- Version using the solvents hot encoded by their ratios.

**Input files:**  polystyrene-imputated-molecular-descriptor-includes-inj-volume-2-15-25.xlsx"

**Picture Output Folders so far:**

1. "Polystyrene-features"
2. "Polystyrene-Hyperopt-Plots"
3. "Polystyrene-Polarity-Plots"

This script takes the preprocessed and imputed polystyrene subset to be used in a XGBoost model. This model is only relevant to binary solvent combinations and is only focused on predicting **only the highest ratio of the binary composition.** The next version aims to move the ratios to an assortment of hot encoded solvent columns. Then the prediction of the ratios will also predict which solvents are used.

I have been playing with tuning lots of XGBoost parameters using Hyperopt within ranges typically used for small datasets. The unfinished version I am trying to implement nested 5-fold cross validation rather than five-fold validation – this would be better for not over fitting the hyper parameters is my understanding.

**Multi-Output-Versions Folder**

These scripts utilize the preprocessing that hot encodes the solvents by their ratios. By predicting the ratios, it is also predicting which solvents are used. Multiple versions are being tested – a multioutput regressor wrapping of the xgboost from scikit learn, meaning a single model outputs all the predicted ratios, compared to versions that have models that predict every column individually.

**Using Scikit learn’s MultiOutputRegressor**

**Script:** XgBoost-SolvHot-Unfinished-MultiOutput-Regressor.ipynb

**Input file:** polystyrene-imputated-solvents-hot-encoded-2-23-25.xlsx

**Output:** Currently no files are saved. Plots are only shown currently.

This is an upgraded version of the XgBoost model that is wrapped with the MultiOutputRegressor to predict all of the solvents’ ratio values at one time. It is very slow to run, and I still need to work on it. Currently, accuracy is worse than if the mean was predicted for all cases, R^2 of negative values. One issue with MultiOutputRegressor is that it can not handle early stopping as a parameter. It does use 5 fold cross validation.

**Single Output Models – 5 Fold Cross Validation used**

**Script:** Xgboost-SolvHot-Unfinished-SingleOutputModels.py

**Input file:** polystyrene-imputated-solvents-hot-encoded-2-23-25.xlsx

**Output:** Currently no files are saved. Plots are only shown currently.

This version creates a model for each solvent column independently but hyperopt tuning is done to find the best hyper parameters with all the models in mind. All of the multioutput models take a while to run, so tuning every model separately could be done but would be very very slow. Note it has a terrible accuracy currently.

**Single Output Models – Nested 5 Fold Cross Validation used**

**Script:** Xgboost-SolvHot-Nested5fold-Unfinished-very-long-runtime.py

**Input file:** polystyrene-imputated-solvents-hot-encoded-2-23-25.xlsx

**Output: “**nested\_cv\_parity\_plots” folder

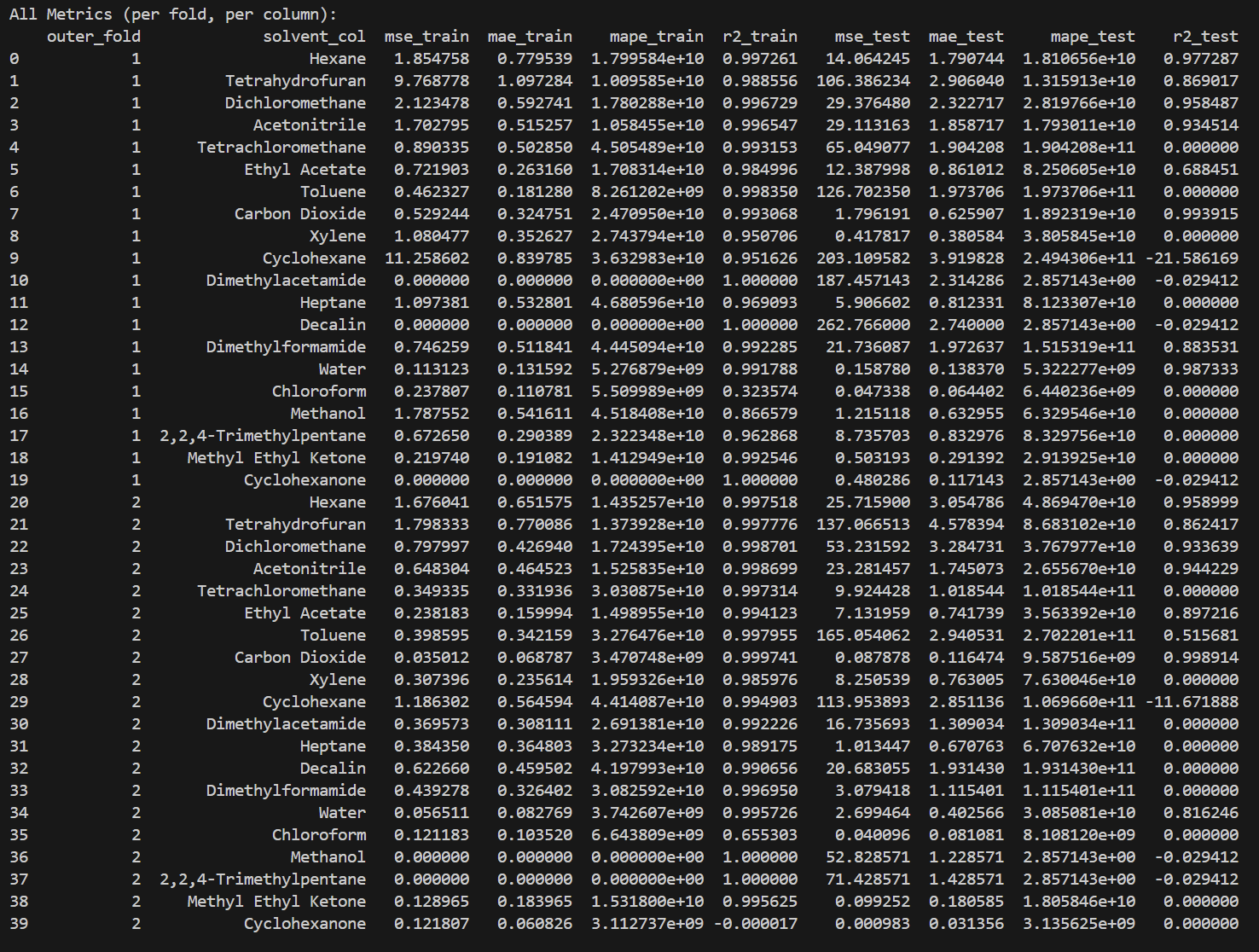
This version is similar to the last version above. However, I am trying to introduce Nested 5 Fold Cross Validation, that I’ve seen is the “gold standard” for validation currently. Hoping it might improve accuracy without the need to do hyperopt tuning for every model separately. On the first run, I am noticing that some solvents seem to have a high accuracy while others perform very poorly. Most likely, the hyper parameters are being tuned strongly toward our more frequent solvent entries.

Attaching screenshot of metrics for first test below

A screenshot of a computer

AI-generated content may be incorrect.

**Single Output Models – Nested 5 Fold Cross Validation used, Continued**

Metrics for first test cont.