Electronic supplementary materials *Mendeleev Commun.*, 2024, **34**, 788–791

Towards machine learning prediction of the fluorescent protein absorption spectra

Roman A. Stepanyuk, Igor V. Polyakov, Anna M. Kulakova, Ekaterina I. Marchenko and Maria G. Khrenova

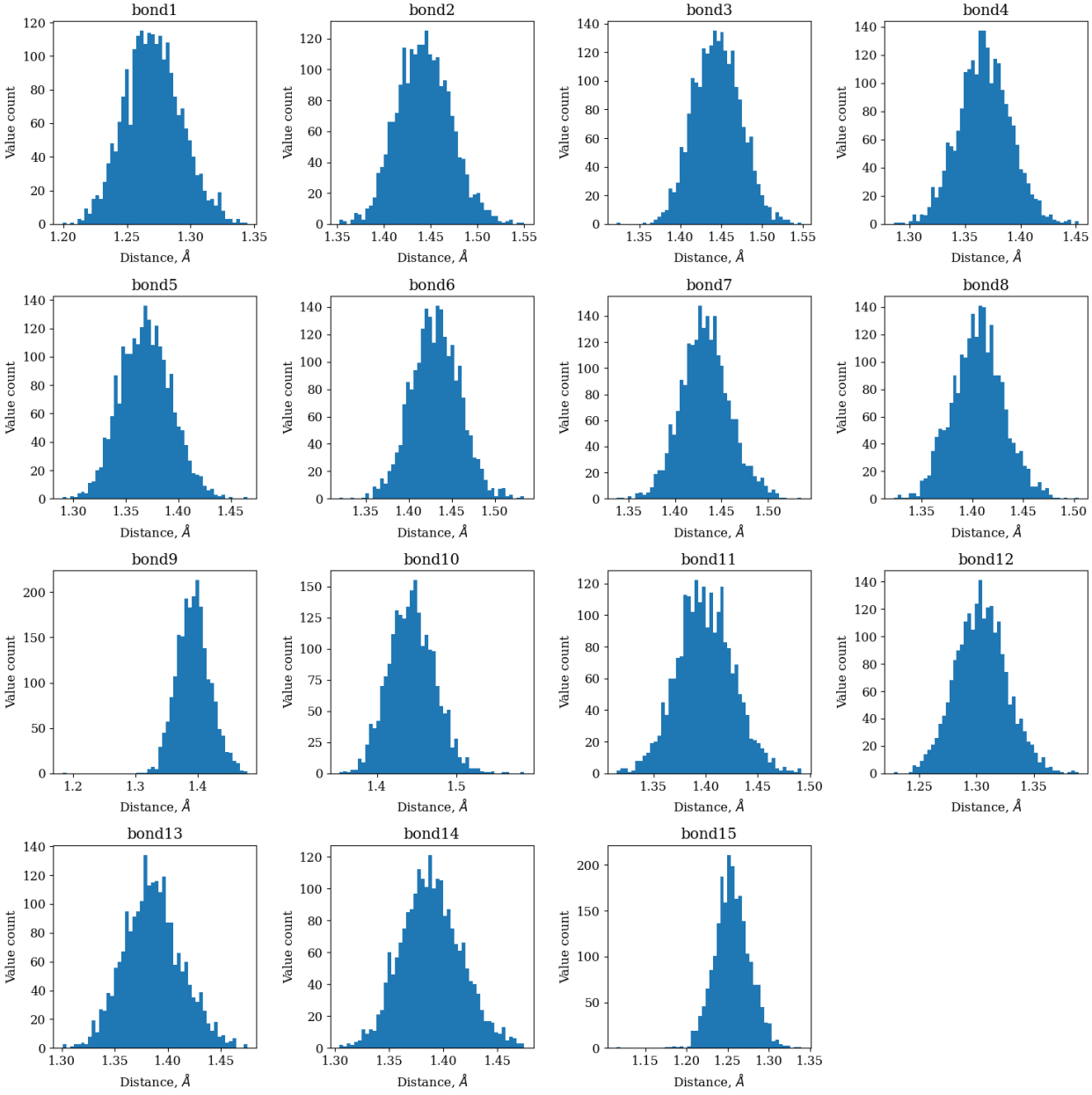

Figure S1. Bonds distributions in dataset.

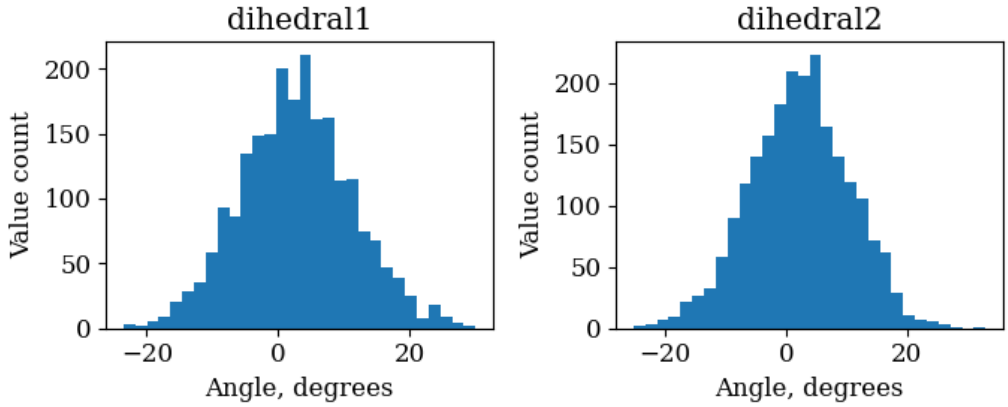

Figure S2. Dihedrals distributions in dataset.

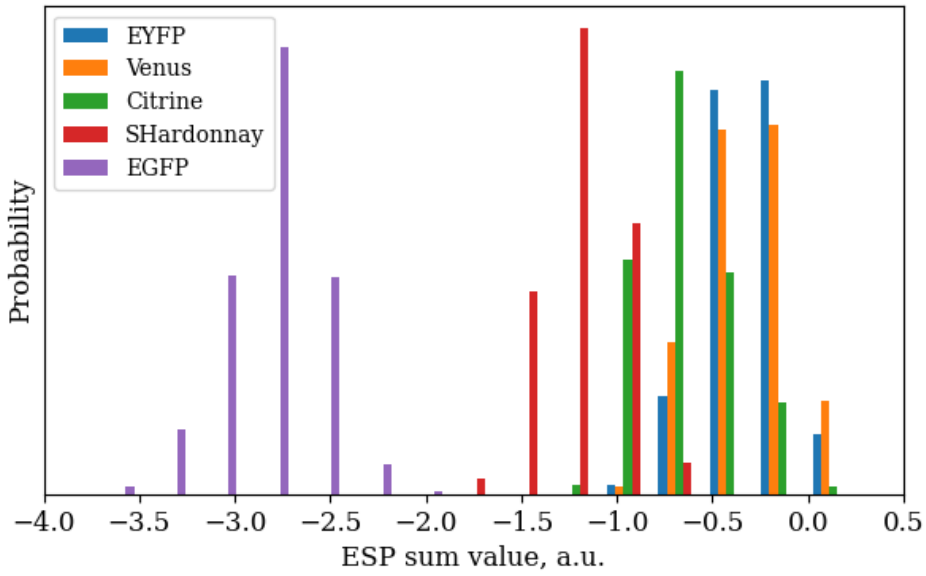

Figure S3. The ESP feature (sum of ESP on all atoms) distribution in dataset.

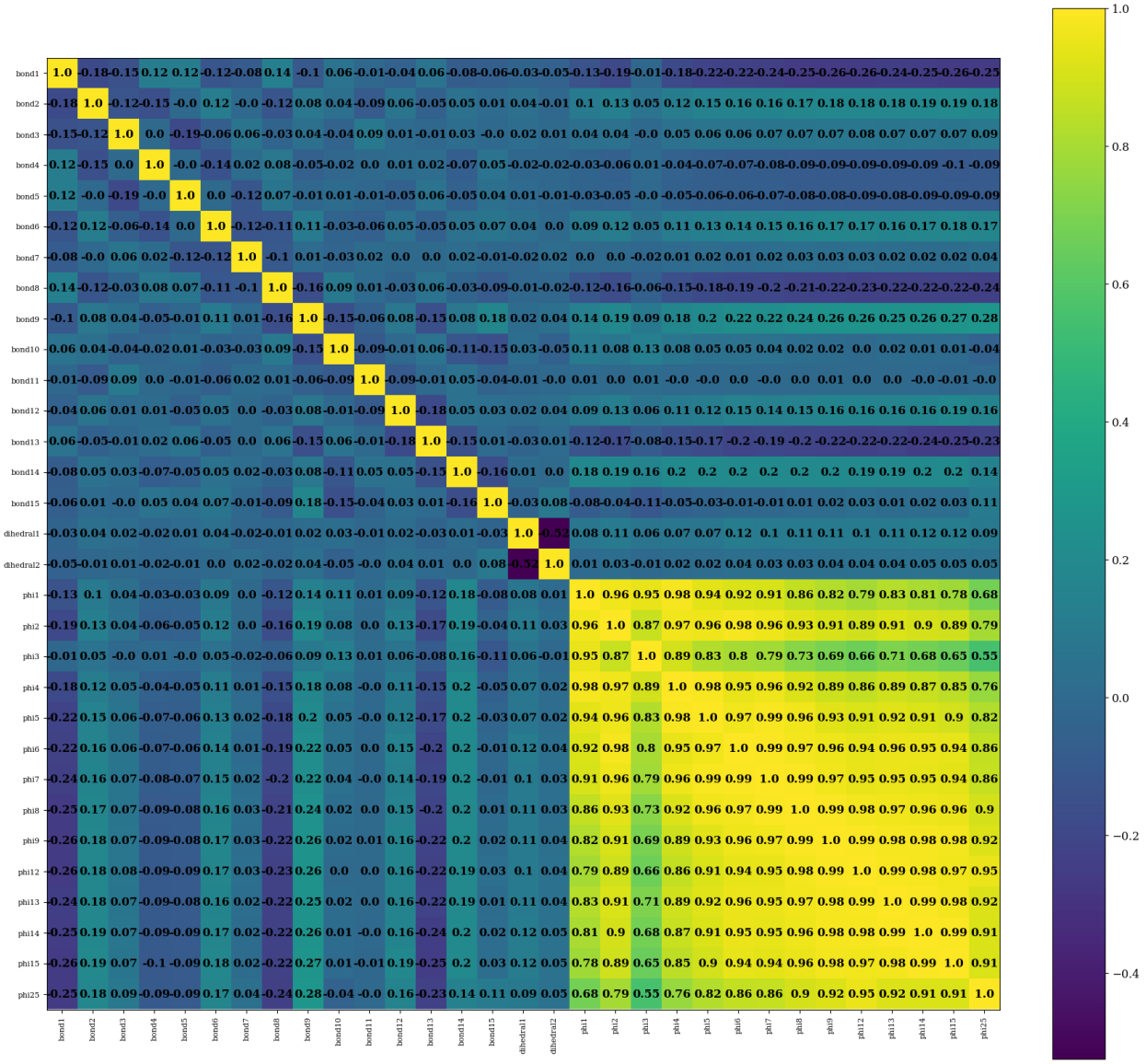

Figure S4. Feature correlation matrix.

not using the esp

```
for prot in prot_list:
    lr_cv_score = cross_val_score(estimator = pile_lr, X= X_train[X_train['prot'] == prot].drop(labels= ['prot', 'esp_sum'],
                                                                                                axis=1),
                y=y_train[X_train['prot'] == prot], n_jobs=-1,
                scoring='neg_mean_absolute_error', cv=4)
    print(prot+' LR 4-fold CV mean:', round(abs(lr_cv_score.mean()),3))
    pile_lr.fit(X_train[X_train['prot'] == prot].drop(labels= ['prot', 'esp_sum'], axis=1),y_train[X_train['prot'] == prot])
    y_pred = pile_lr.predict(X_test[X_test['prot'] == prot].drop(labels= ['prot', 'esp_sum'], axis=1))
    print(prot+' LR test MAE:', round(mean_absolute_error(y_test[X_test['prot'] == prot], y_pred),3))
    print(prot+' LR test R2:', round(r2_score(y_test[X_test['prot'] == prot], y_pred),3))
```

```
eyfp LR 4-fold CV mean: 0.288
eyfp LR test MAE: 0.251
eyfp LR test R2: 0.723
venus LR 4-fold CV mean: 0.323
venus LR test MAE: 0.327
venus LR test R2: 0.462
citrin LR 4-fold CV mean: 0.247
citrin LR test MAE: 0.22
citrin LR test R2: 0.706
shardonnay LR 4-fold CV mean: 0.295
shardonnay LR test MAE: 0.295
shardonnay LR test R2: 0.189
egfp LR 4-fold CV mean: 0.138
egfp LR test MAE: 0.123
egfp LR test R2: 0.935
```

with the esp

```
for prot in prot_list:
    lr_cv_score = cross_val_score(estimator = pile_lr, X= X_train[X_train['prot'] == prot].drop(labels= ['prot'], axis=1),
                y=y_train[X_train['prot'] == prot], n_jobs=-1,
                scoring='neg_mean_absolute_error', cv=4)
    print(prot+' LR 4-fold CV mean:', round(abs(lr_cv_score.mean()),3))
    pile_lr.fit(X_train[X_train['prot'] == prot].drop(labels= ['prot'], axis=1),y_train[X_train['prot'] == prot])
    y_pred = pile_lr.predict(X_test[X_test['prot'] == prot].drop(labels= ['prot'], axis=1))
    print(prot+' LR test MAE:', round(mean_absolute_error(y_test[X_test['prot'] == prot], y_pred),3))
    print(prot+' LR test R2:', round(r2_score(y_test[X_test['prot'] == prot], y_pred),3))
```

```
eyfp LR 4-fold CV mean: 0.285
eyfp LR test MAE: 0.252
eyfp LR test R2: 0.725
venus LR 4-fold CV mean: 0.323
venus LR test MAE: 0.328
venus LR test R2: 0.459
citrin LR 4-fold CV mean: 0.249
citrin LR test MAE: 0.214
citrin LR test R2: 0.716
shardonnay LR 4-fold CV mean: 0.296
shardonnay LR test MAE: 0.295
shardonnay LR test R2: 0.193
egfp LR 4-fold CV mean: 0.139
egfp LR test MAE: 0.121
egfp LR test R2: 0.937
```

Figure S5. Individual linear models result.

```
lr_coef_df = pd.DataFrame({"Feature":X_train.drop(labels= ['prot'], axis=1).columns.tolist(),
                           "Coefficients":pile_lr.named_steps['model'].coef_})
lr_coef_df['abs_coef'] = lr_coef_df['Coefficients'].astype('float').abs()
lr_coef_df.sort_values(by='abs_coef', ascending=False)
```

	Feature	Coefficients	abs_coef
7	bond8	0.319883	0.319883
8	bond9	-0.261963	0.261963
5	bond6	-0.162945	0.162945
6	bond7	-0.145625	0.145625
9	bond10	0.130937	0.130937
3	bond4	0.105173	0.105173
10	bond11	0.099151	0.099151
4	bond5	0.070345	0.070345
14	bond15	-0.066325	0.066325
11	bond12	-0.065213	0.065213
0	bond1	0.051899	0.051899
1	bond2	-0.050590	0.050590
12	bond13	0.049666	0.049666
2	bond3	-0.048492	0.048492
13	bond14	-0.033273	0.033273
15	dihedral1	0.024475	0.024475
17	esp_sum	-0.018151	0.018151
16	dihedral2	-0.009225	0.009225

Figure S6. Feature importance analysis for linear model.