

Predicting COVID Moonshot molecule binding affinity



Isaac Ellmen & Sim Zhao
University of Oxford

1. Determine binding affinity of putative SARS-CoV-2 protease binders
2. Strong focus on workflow automation
3. Snakemake powers the entire process

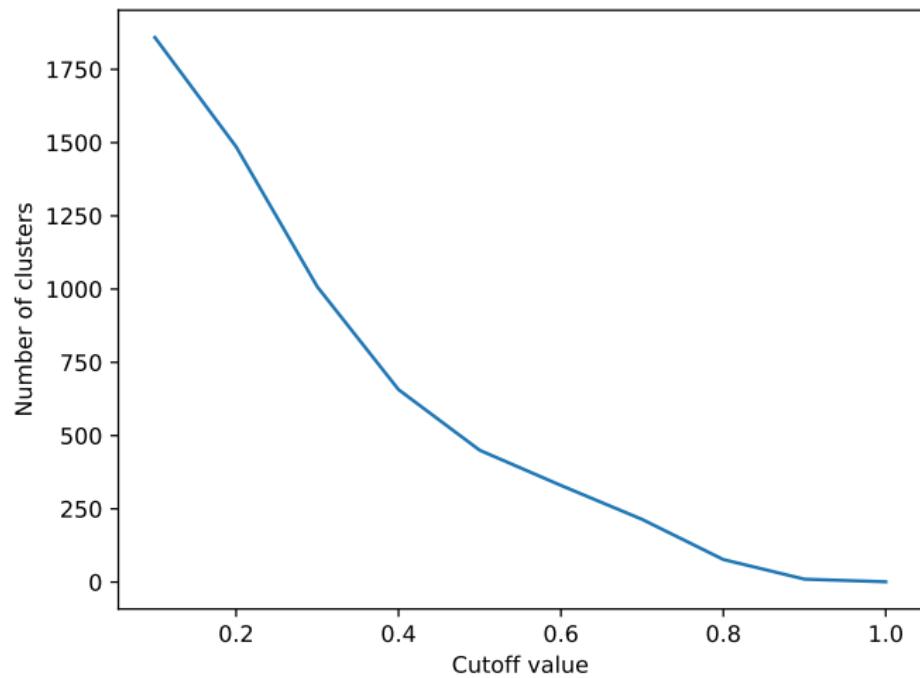
Our workflow DAG



Build database

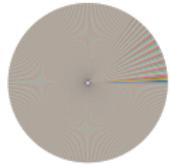
```
1 import sqlite3
2 import pandas as pd
3
4 def build_db(data_path, out_path):
5     con = sqlite3.connect(out_path)
6     cur = con.cursor()
7     df = pd.read_csv(data_path, index_col='CID')
8     assays_df = df.drop(columns=['SMILES'])
9     compounds_df = df[['SMILES']]
10    assays_table = 'assays'
11    assays_df.to_sql(assays_table, con, if_exists='replace', index=True)
12    compounds_table = 'compounds'
13    compounds_df.to_sql(compounds_table, con,
14                         if_exists='replace', index=True)
14    con.close()
```

Fingerprint clustering - sensitivity to cutoff

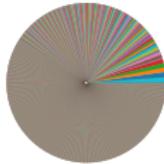


Fingerprint clustering - class composition

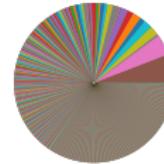
Size distribution of clusters
1859 clusters generated with a cutoff of 0.1



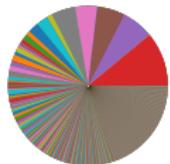
Size distribution of clusters
1486 clusters generated with a cutoff of 0.2



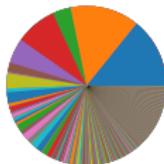
Size distribution of clusters
1008 clusters generated with a cutoff of 0.3



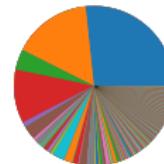
Size distribution of clusters
657 clusters generated with a cutoff of 0.4



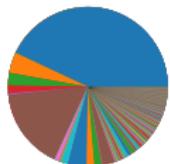
Size distribution of clusters
450 clusters generated with a cutoff of 0.5



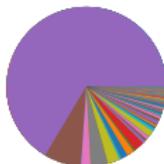
Size distribution of clusters
330 clusters generated with a cutoff of 0.6



Size distribution of clusters
214 clusters generated with a cutoff of 0.7



Size distribution of clusters
77 clusters generated with a cutoff of 0.8

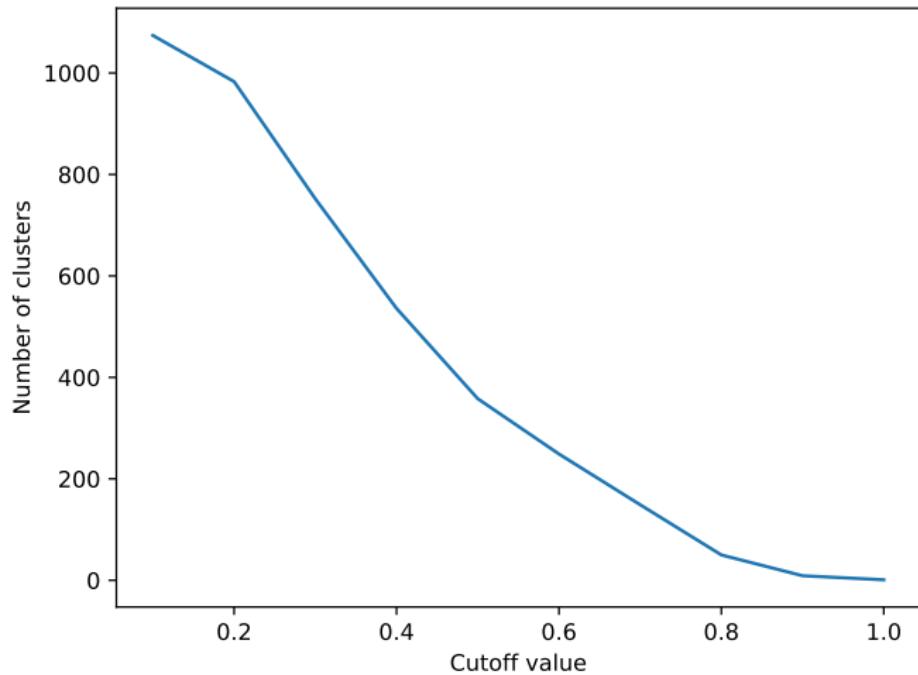


Size distribution of clusters
10 clusters generated with a cutoff of 0.9



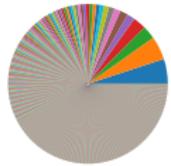
Scaffold clustering - sensitivity to cutoff

7

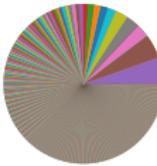


Scaffold clustering - class composition

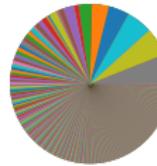
Size distribution of clusters
1074 clusters generated with a cutoff of 0.1



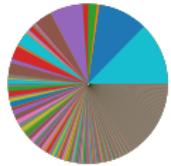
Size distribution of clusters
983 clusters generated with a cutoff of 0.2



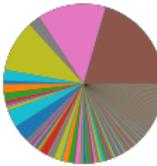
Size distribution of clusters
752 clusters generated with a cutoff of 0.3



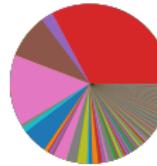
Size distribution of clusters
536 clusters generated with a cutoff of 0.4



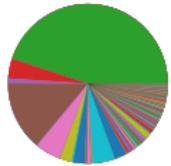
Size distribution of clusters
358 clusters generated with a cutoff of 0.5



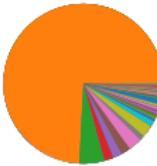
Size distribution of clusters
249 clusters generated with a cutoff of 0.6



Size distribution of clusters
149 clusters generated with a cutoff of 0.7

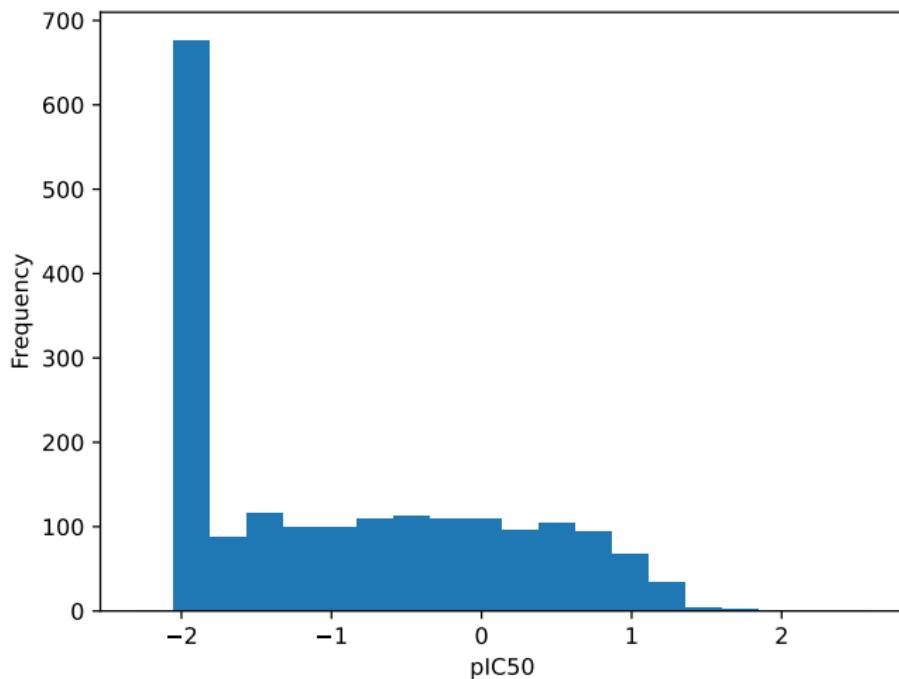


Size distribution of clusters
50 clusters generated with a cutoff of 0.8



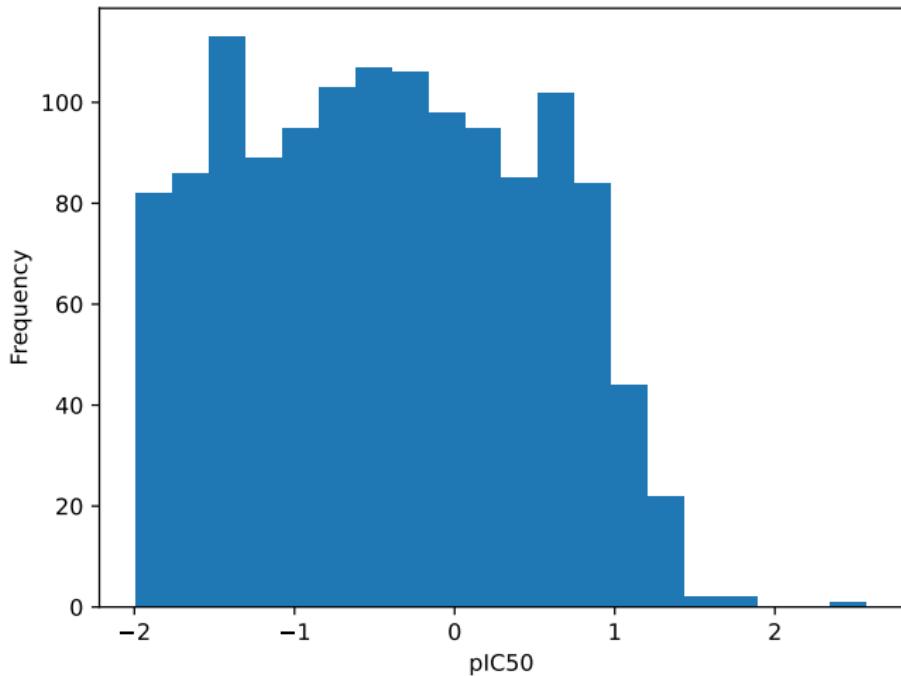
Size distribution of clusters
9 clusters generated with a cutoff of 0.9





pIC50 distribution (only binders)

10



1. Classification for binders vs. non-binders
2. Regression for binding strength (pIC50)
3. Can use similar model for both since learning very similar properties

Compute Descriptors

```
1 def compute_descriptors(molecule):
2     descriptors = {}
3     for d in Descriptors.descList:
4         try:
5             value = d[1](molecule)
6             descriptors[d[0]] = value
7
8         except:
9             value = np.nan
10            descriptors[d[0]] = np.nan
11    descriptors = pd.Series(descriptors)
12    return descriptors
```

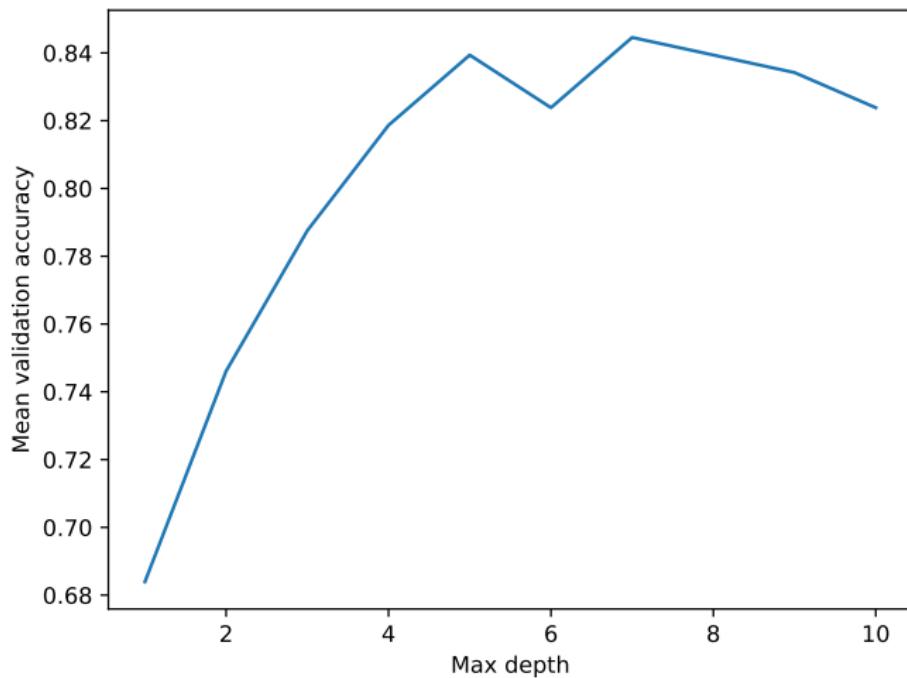
Make Dataframe

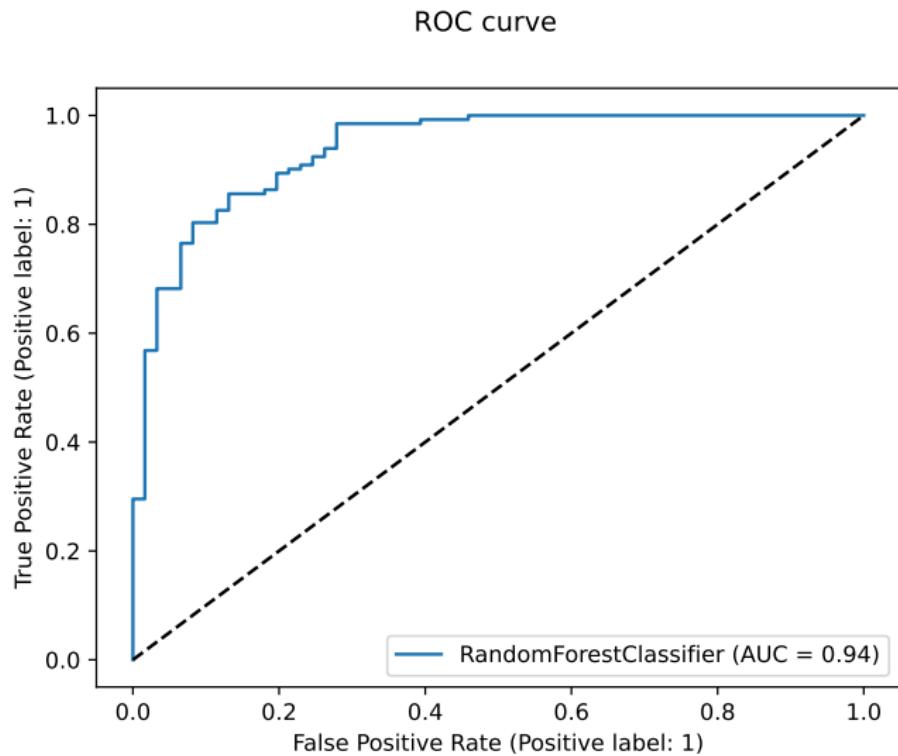
```
1 def pack_df(df_with_SMILES):
2     descriptorsdf = pd.DataFrame()
3     for index, row in df_with_SMILES.iterrows():
4         # Get the SMILES string of the compound
5         smiles = row['SMILES']
6         # Create a rdkit molecule object from SMILES
7         mol = Chem.MolFromSmiles(smiles)
8         des = compute_descriptors(mol)
9         for i in range(len(des)):
10             descriptorsdf.at[index, '{}'.format(
11                 des.index[i])] = des.values[i]
11 return descriptorsdf
```

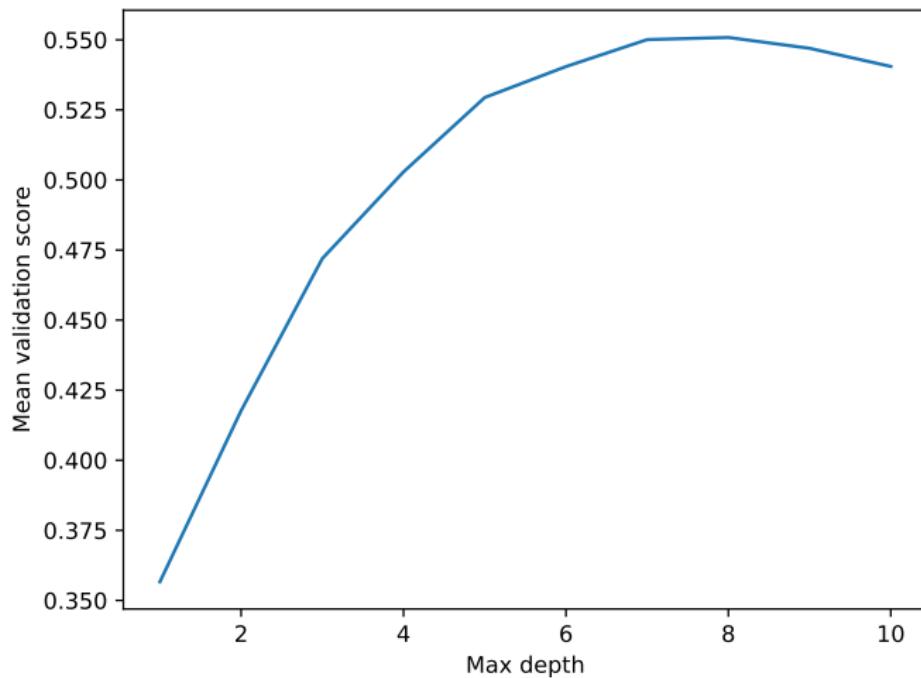
Feature Selection

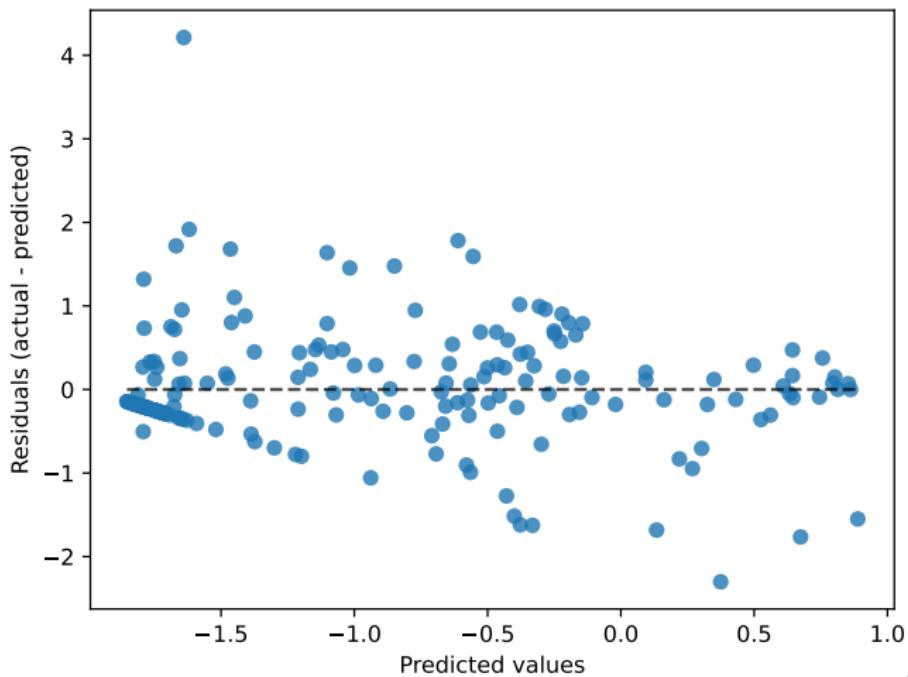
```
1 def feature_selection(Descriptors_df, y_df, k_feature
2     = 10, task = 'regress', way = 'simple', rfe_step
3     = 0.2):
4     if task == 'regress':
5         estimator = SVR(kernel="linear")
6     if task == 'classification':
7         estimator = LogisticRegression()
8     if way == 'simple':
9         selector = SelectKBest(f_classif, k=k_feature
10                         )
11     if way == 'rfe':
12         selector = RFE(estimator,
13                         n_features_to_select=k_feature, step=
14                         rfe_step)
15     selector.fit(Descriptors_df, y_df)
16     top_k_idx = selector.get_support(indices=True)
17     return Descriptors_df.columns[top_k_idx]
```

Feature Descriptor	Meaning
PEOE_VSA8	Polarizability based on Eigenvalues of the Overlap matrix (PEOE) with 8 VSA
fr_N_O	Nitro group
fr_diazo	Diazo group
fr_dihydropyridine	Dihydropyridine group
fr_hdrzine	Hydrazine group
fr_ketone_Topliss	Ketone group using Topliss method
fr_lactam	Lactam group
fr_oxazole	Oxazole group
fr_sulfonamid	Sulfonamide group
fr_sulfone	Sulfone group









1. Trained a classifier and a regressor for binding affinity
2. Automated entire pipeline, from raw data to slides
3. Need for standardization of data, metrics