Protein Covariance:
Predicting Phenotypes Based On Amino Acid Sequences

Lara Gokcelioglu
Institute For Computing In Research Santa Fe

July 30, 2024

Project Overview

Mix of **biology and computing**

Phenotypes of proteins **based on their amino acid sequences**.

6 different models **for 3 different phenotypes and 2 representations each**.

Dataset

The dataset consists of **227 proteins** and each protein's **amino acid sequence**–which later gets **fixed to 512 positions.**

division	organism_id	ex_max	em_max	pdb_0	seq_length	seq
other sequences	32630	342	382	NaN	30	ELSKETALKKSFKFLVLIILWNNTVDAIHI
hydrozoans	6100	355	424	NaN	239	MVSKGEELFTGVVPILVELDGDVNGHRFSVSGEGEGDATYGKLTLK...
sea anemones	475174	375	458	NaN	227	MAGLLKESMRIKMDMEGTVNGHYFKCEGEGDGNPFTGTQSMRIHVT...
hydrozoans	6100	379	446	NaN	239	MVSKGEELFTGVVPILVELDGDVNGHKFSVRGEGEGDATNGKLTLK...
hydrozoans	6100	380	446	NaN	239	MVSKGEELFTGVVPILVELDGDVNGHKFSVSGEGEGDATYGKLTLK...
...	...	...	...	...	...	...
bacteria	1299	697	720	NaN	320	MSRDPLPFFPPLYLGGPEITTENCEREPIHIPGSIQPHGALLTADG...
a-proteobacteria	1076	700	719	NaN	316	MAEGSVARQPDLLTCDDEPIHIPGAIQPHGLLLALAADMTIVAGSD...
bacteria	1299	701	719	3S7Q	335	MASMTGGQQMGRGSMSRDPLPFFPPLYLGGPEITTENCEREPIHIP...
a-proteobacteria	1076	701	720	NaN	316	MAEGSVARQPDLLTCDDEPIHIPGAIQPHGLLLALAADMTIVAGSD...
a-proteobacteria	1076	702	720	NaN	316	MAEGSVARQPDLLTCDDEPIHIPGAIQPHGLLLALAADMTIVAGSD...

Figure: Partial Dataset

Phenotypes

The first phenotype is **em_max**

The second phenotype is **ex_max**

The third and last phenotype is **states_0_brightness**

Representations

There are two representations of the data:

pc_coords

proteins_projected_pc_coords

Data Manipulation

aminoacids

leftjustified_seqs

match_aminoacids

Singular Value Decomposition

Aminoacids

```
# array of amino acids in a certain protein ( sirius aequorea victoria row(1) )
aminoacids = set(df.seq[0])   ###sets the aminoacids variable to the 1 row of seq column
aminoacids.add('*')
aminoacids = np.array(list(aminoacids))   ###turns the 1 line of seq column into a python list
aminoacids.sort()
aminoacids # all amino acids plus termination
```

✓ 0.0s Python

```
array(['*', 'A', 'D', 'E', 'F', 'H', 'I', 'K', 'L', 'N', 'S', 'T', 'V',
       'W'], dtype='<U1')
```

Figure: aminoacids Variable

leftjustified_seqs

```
### generating matrices
#                      add termination char        left justify    split all chars  to numpy array
leftjustified_seqs = (df.seq.astype(str) + "*").str.ljust(512, " ").apply(list).apply(np.array)
# vertically concatenate all proteins
leftjustified_seqs = np.vstack(leftjustified_seqs)

###left_justified_seqs are fixed to the size of 512 with this function
leftjustified_seqs
```

✓ 0.1s Python

```
array([['E', 'L', 'S', ..., ' ', ' ', ' '],
       ['M', 'V', 'S', ..., ' ', ' ', ' '],
       ['M', 'A', 'G', ..., ' ', ' ', ' '],
       ...,
       ['M', 'A', 'S', ..., ' ', ' ', ' '],
       ['M', 'A', 'E', ..., ' ', ' ', ' '],
       ['M', 'A', 'E', ..., ' ', ' ', ' ']], dtype='<U1')
```

Figure: leftjustified_seqs variable

match_amoinoacids Function

```
def match_aminoacids(protein_sequence, chars=aminoacids):
    """
    compares each pair of amino-acid at position in protein against
    our library of amino-acids that are currently set to line 1 of the seq column of the dataframe
    (returns one hot)
    """
    return np.equal.outer(protein_sequence, chars).astype(np.float64)

match_aminoacids
```

✓ 0.0s Python

```
<function __main__.match_aminoacids(protein_sequence, chars=array(['*', 'A', 'D', 'E', 'F', 'H', 'I', 'K', 'L', 'N
       'W'], dtype='<U1'))>
```

Figure: match_aminoacids function

Singular Value Decomposition

```
u, s, vt = np.linalg.svd(protein_ohe)
# matrix multiplication (this is just scaling each left singular vector, by it singular value)
pc_coords = u @ np.diag(s)

✓ 22.1s                                                                    Python
```

Figure: Application Of SVD

Training The Model

K-Fold Validation

splits k=10

90% train set vs. 10% test set

```
import numpy as np
from sklearn.model_selection import KFold

X = pc_coords
y = (df["states_0_brightness"])

kf = KFold(n_splits=10)

for train_index, test_index in kf.split(X):
    print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
```

```
###predicting r2 values

from sklearn import datasets, linear_model
from sklearn.model_selection import cross_val_predict


lasso = linear_model.Lasso()
lasso.fit(X_train,y_train)
y_pred = lasso.predict(X_test)
rsq = r2_score(y_test, y_pred)
print("test set:",rsq)
score = lasso.score(X_train,y_train)
print("training set:" , score)
```

K-Fold Cross Validation

Increased **accuracy**

More data for both sets

Reduces variance

R^2 Value

The strength of the **relationship between the model and the dependent variable**

$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}}$$

Coefficient of determination

Analysis Of Em_Max

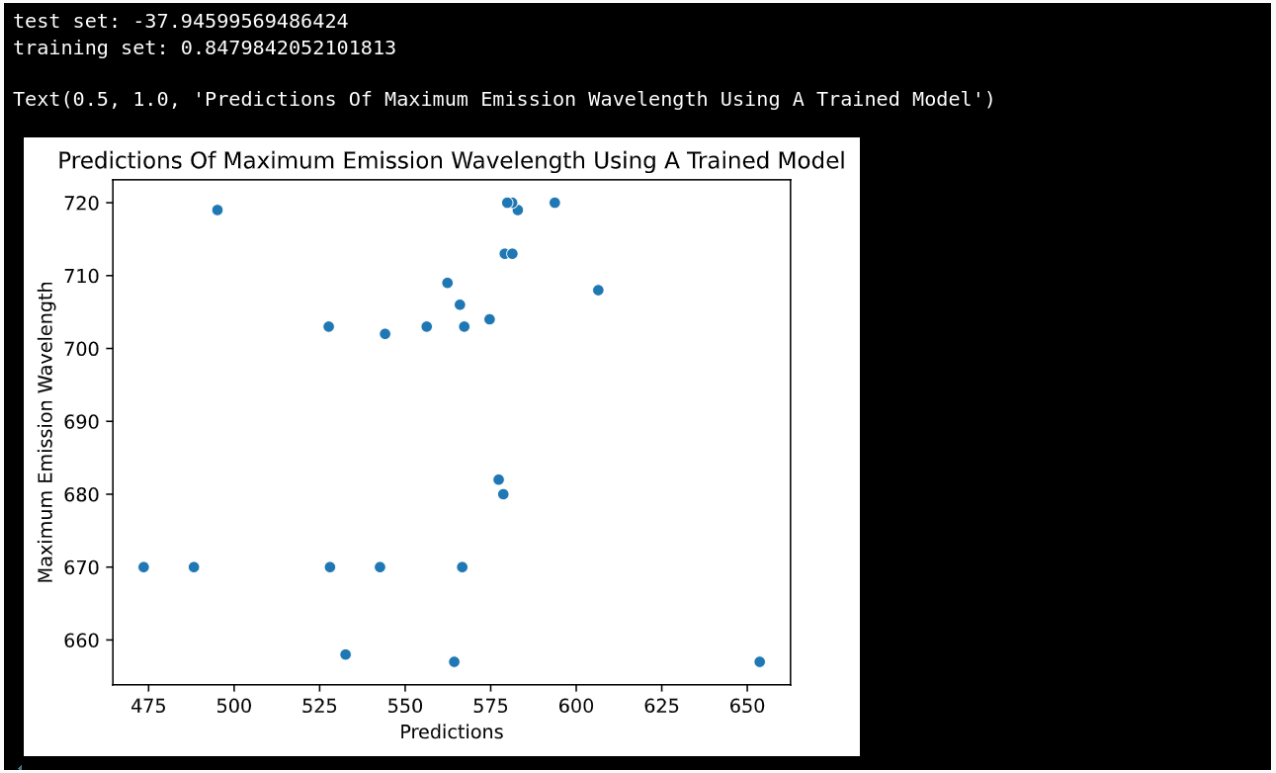


Figure: Analysis Of Most Efficient Emission Wavelength

Analysis Of Projected Em_Max

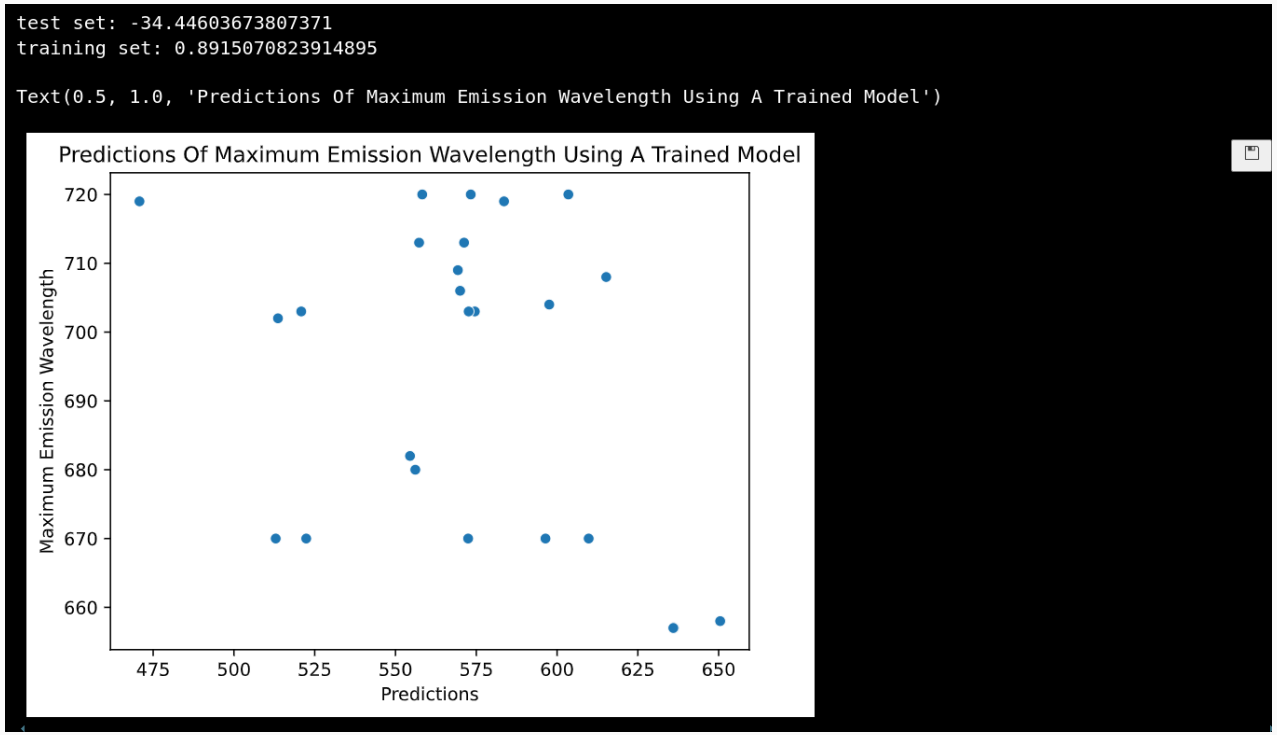


Figure: Analysis Of Most Efficient Emission Wavelength In Projected Representation

Analysis Of Ex_Max

```
test set: -26.921630644805816
training set: 0.8226919339817478

Text(0.5, 1.0, 'Predictions Of Maximum Excitation Wavelength Using A Trained Model')
```

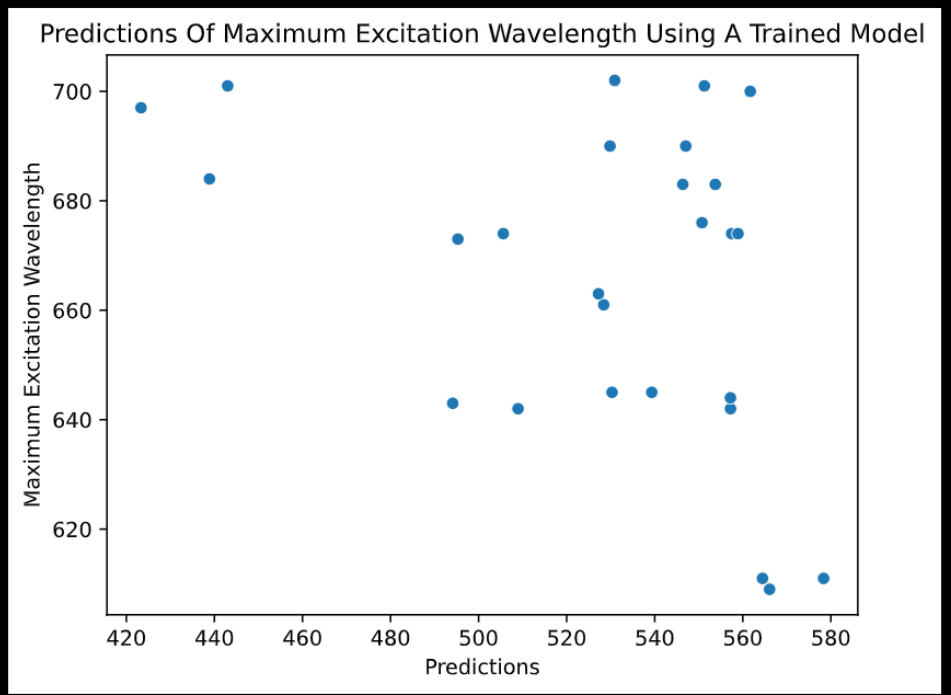


Figure: Analysis Of Most Efficient Excitation Wavelength

Analysis Of Projected Ex_Max

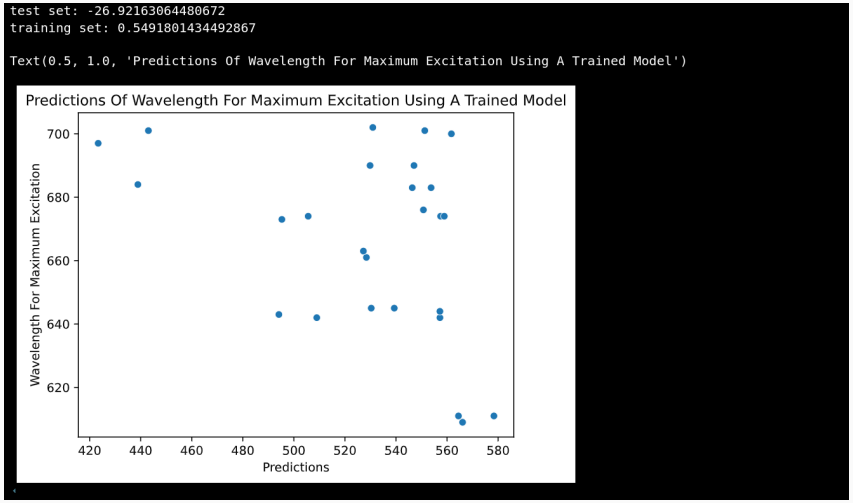


Figure: Analysis Of Most Efficient Excitation Wavelength In Projected Representation

Analysis Of States_0_Brightness

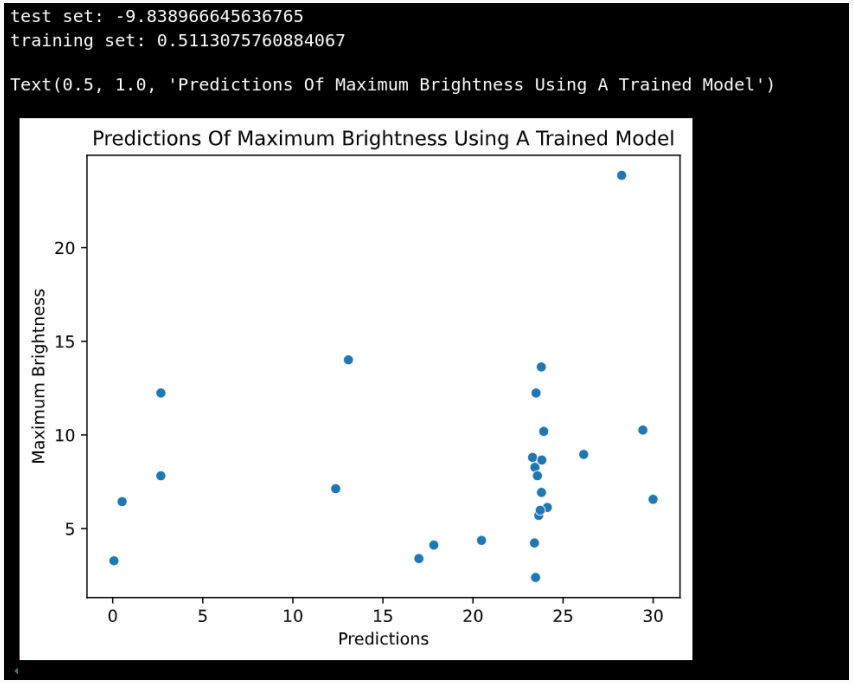


Figure: Analysis of Brightness

Analysis Of Projected States_0_Brightness

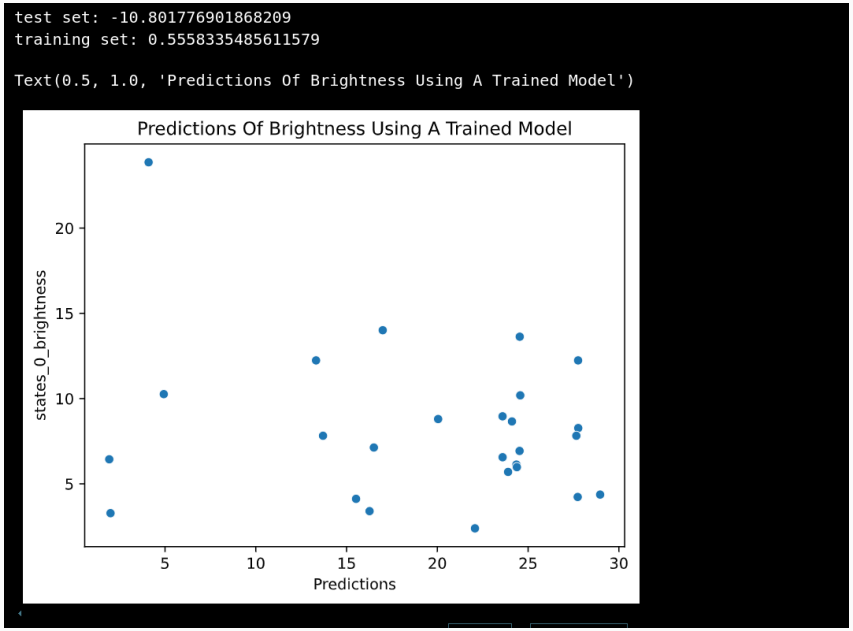
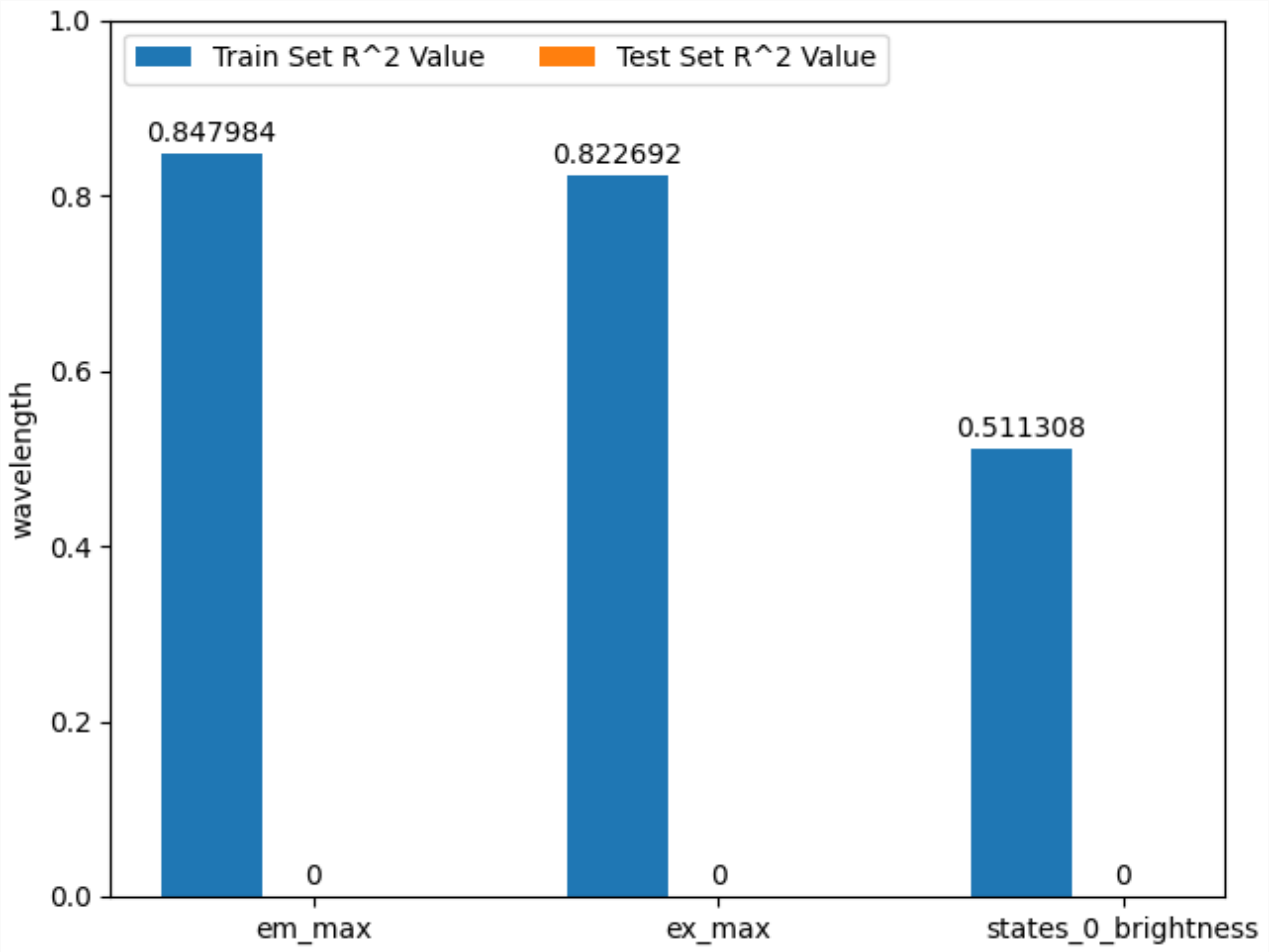
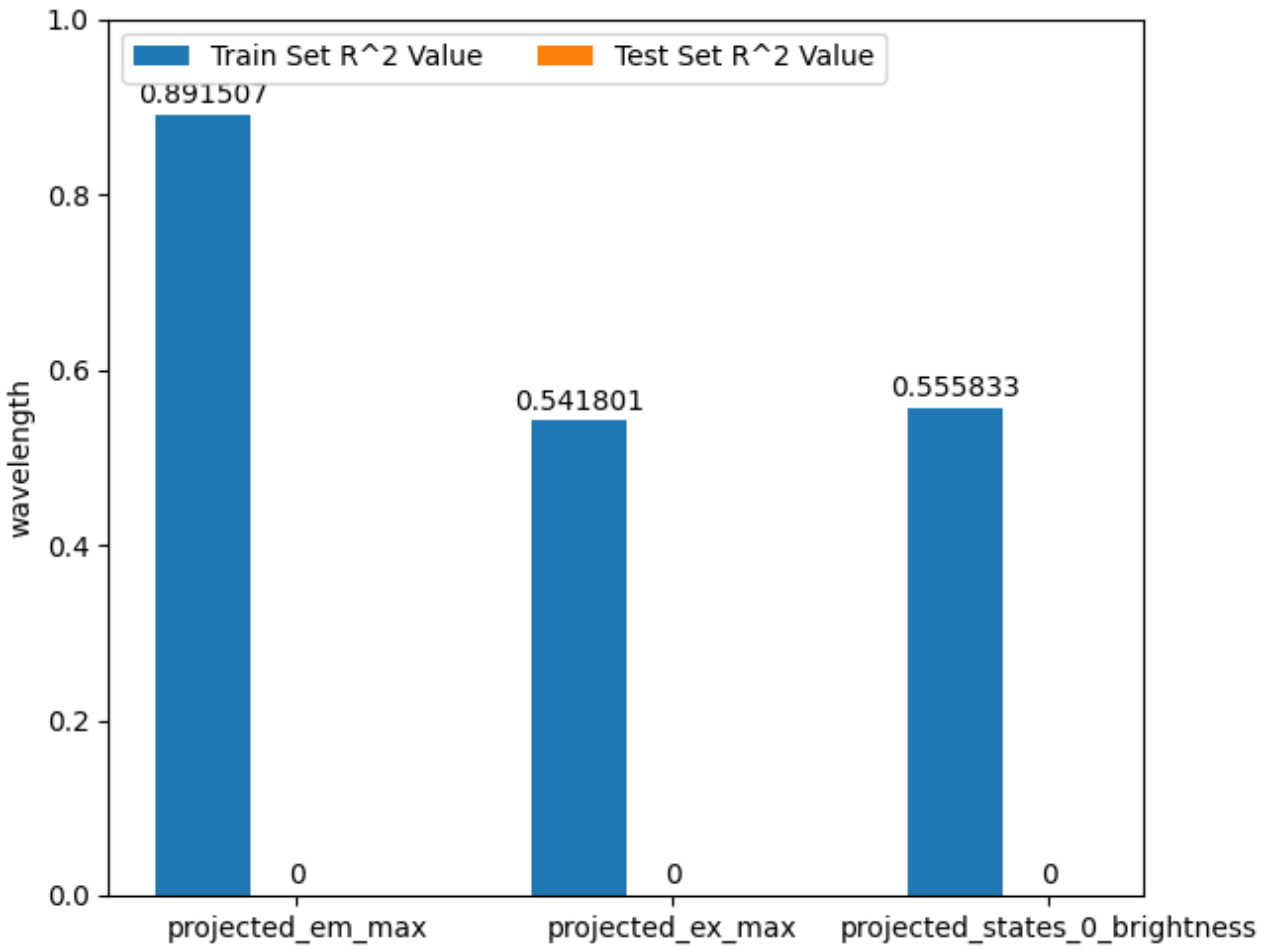


Figure: Analysis of Brightness In Projected Representation

pc_coords R^2 Values



proteins_projected_pc_coords R^2 Values



Future Path

Considering more folds in analysis

Moving beyond linear models

Including more parameters

Challenges

Choosing the best method to train a model on

Lack of answers

Inaccurate results

References

(N.d.). Statisticsbyjim.com. Retrieved July 28, 2024, from https://statisticsbyjim.com/regression/interpret-r-squared-regression/