BIOSOC PROJECTS

DRUG DISCOVERY

ENDTERM EVALUATION
GROUP - 2

MENTORS
NISCHAY PATEL AND VAMSI

GROUP MEMBERS
JAIVEER SABHARWAL
HIMANSHU DUGAR
ENOCH EMMANUEL
ASMITA PAUL

CONTENT

DRUG DISCOVERY

- The aim of the project was to explore the realm of Drug Discovery using machine learning techniques.
- Drug discovery encompasses various methods and procedures of which this project consists of only a small introductory step of Drug-Protein interaction.
- We referred to research papers published on the topic and attempted to **understand, implement, improve and compile** our results.
- Some of our experimentation did not match with the papers we referred to, we have highlighted the possible reasons for the same and potential areas for improvement.

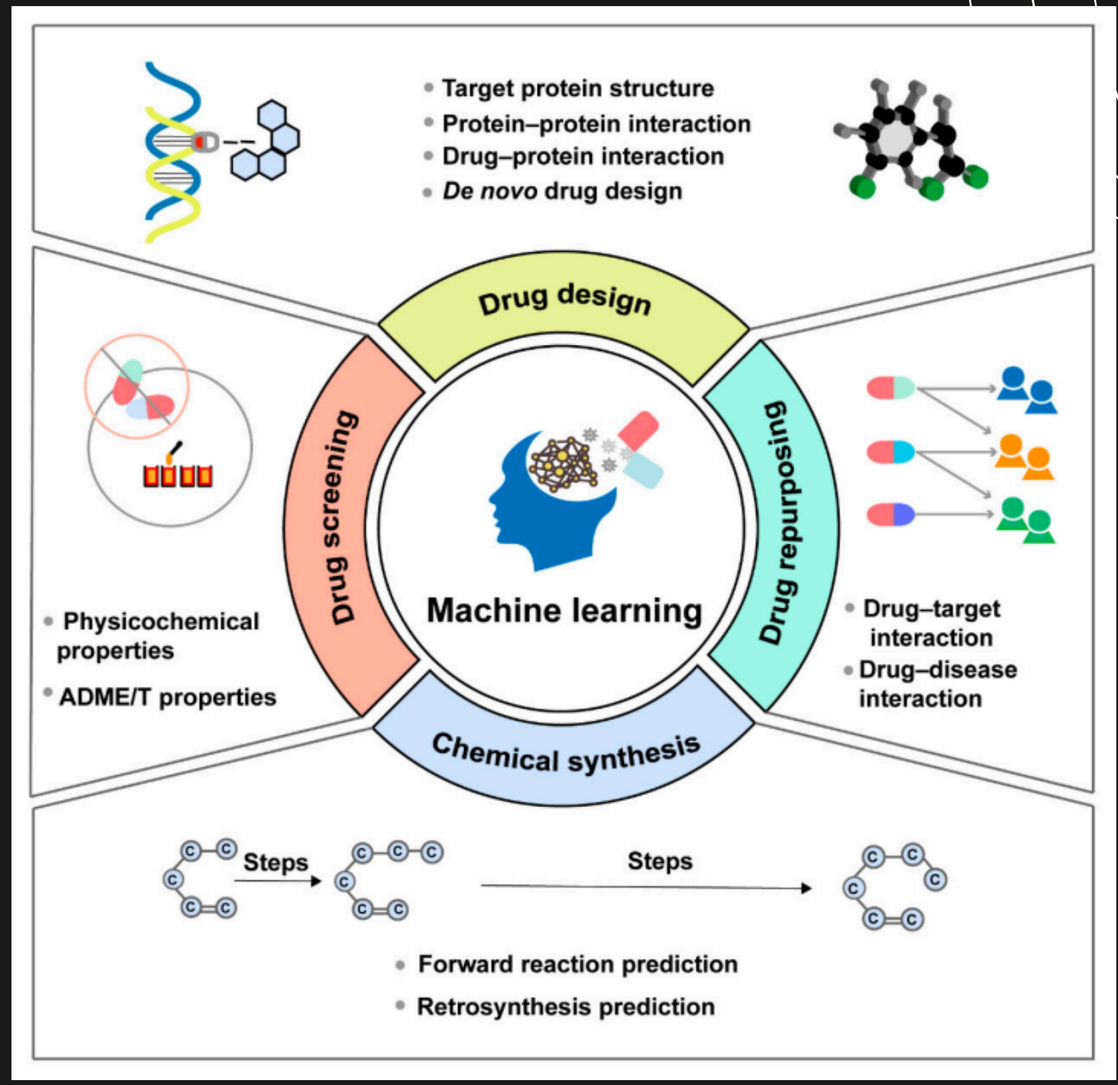
DIGRESSING THE PROBLEM

Drug Discovery consists of 4 major parts:

- Drug Design
- Drug Screening
- Drug Repurposing
- Chemical Synthesis.

While all the parts are crucial, we focus upon the design of the drug itself, particularly drug protein interaction. The availability of datasets pertaining to the same aided our approach.

Drug-Target/Protein interaction datasets are available on the ChEMBL database.



DIGRESSING THE PROBLEM

- The ChEMBL Database is a database that contains curated bioactivity data of more than 2 million compounds. It is compiled from more than 76,000 documents and 1.2 million assays.
- Our desired inputs are molecule structures and for an output we want to know how effective would that molecule be as a drug, for a particular protein.

- The ChEMBL database contains data about thousands of proteins and their interaction with various molecules.

DIGRESSING THE PROBLEM

Here is a ChEMBL database search for the word ‘coronavirus’. We chose to work with the ‘SARS coronavirus 3C-like proteinase’ and the H3N2 organism not shown here.

	cross_references	organism	pref_name	score	species_group_flag	target_chembl_id	target_components	target_type	tax_id
0	[]	Coronavirus	Coronavirus	17.0	False	CHEMBL613732	[]	ORGANISM	11119
1	[]	Feline coronavirus	Feline coronavirus	14.0	False	CHEMBL612744	[]	ORGANISM	12663
2	[]	Murine coronavirus	Murine coronavirus	14.0	False	CHEMBL5209664	[]	ORGANISM	694005
3	[]	Canine coronavirus	Canine coronavirus	14.0	False	CHEMBL5291668	[]	ORGANISM	11153
4	[]	Human coronavirus 229E	Human coronavirus 229E	13.0	False	CHEMBL613837	[]	ORGANISM	11137
5	[]	Human coronavirus OC43	Human coronavirus OC43	13.0	False	CHEMBL5209665	[]	ORGANISM	31631
6	[{'xref_id': 'P0C6U8', 'xref_name': None, 'xre...	SARS coronavirus	SARS coronavirus 3C-like proteinase	10.0	False	CHEMBL3927	[{'accession': 'P0C6U8', 'component_descriptio...	SINGLE PROTEIN	227859

WORKFLOW

- Query the ChEMBL database for H3N2 and coronavirus, obtain a suitable dataset for drug interaction with the two.
- Expand the dataset using feature engineering - Molecular and Lipinski descriptors.
- Select the required information and perform exploratory data analysis (EDA).
- Process the data and further cleaning up using algorithms like IWSSR.
- Apply various models and tune hyperparameters to achieve better metrics.
- Compare all model parameters and compile results.

FINAL COMMENTS

- The only information we have initially is the structure of the molecule - canonical_smiles and the effectiveness of it as a drug on coronavirus.
- We must use feature engineering to add numerical features which describe the molecule and its effect - descriptors.

	molecule_chembl_id	canonical_smiles	standard_value
0	CHEMBL187579	Cc1noc(C)c1CN1C(=O)C(=O)c2cc(C#N)ccc21	7200.0
1	CHEMBL188487	O=C1C(=O)N(Cc2ccc(F)cc2Cl)c2ccc(I)cc21	9400.0
2	CHEMBL185698	O=C1C(=O)N(CC2COc3ccccc3O2)c2ccc(I)cc21	13500.0
3	CHEMBL426082	O=C1C(=O)N(Cc2cc3ccccc3s2)c2ccccc21	13110.0
4	CHEMBL187717	O=C1C(=O)N(Cc2cc3ccccc3s2)c2c1cccc2[N+](=O)[O-]	2000.0
...	...	...	...
109	CHEMBL2146517	COC(=O)[C@@]1(C)CCCc2c1ccc1c2C(=O)C(=O)c2c(C)c...	21100.0
110	CHEMBL187460	C[C@H]1COC2=C1C(=O)C(=O)c1c2ccc2c1CCCC2(C)C	226700.0
111	CHEMBL363535	Cc1coc2c1C(=O)C(=O)c1c-2ccc2c(C)cccc12	38700.0
112	CHEMBL227075	Cc1cccc2c3c(ccc12)C1=C(C(=O)C3=O)[C@@H](C)CO1	14400.0
113	CHEMBL45830	CC(C)C1=Cc2ccc3c(c2C(=O)C1=O)CCCC3(C)C	21100.0

86 rows × 3 columns

- After adding numerical features, the problem boils down to being a regression problem - one where we can predict the output value from a pre-trained model when the structure of a molecule is passed.

- We used Conda and rdkit to set up the environment. Using the RDKit molecule objects on canonical SMILES , various Lipinski descriptors are calculated.

	molecule_chembl_id	canonical_smiles	standard_value	class	MW	LogP	NumHDonors	NumHAcceptors
0	CHEMBL502613	CC(=O)N[C@H]1[C@H]([C@H](O)[C@H](O)CO)O[C@](OC...	3000.0	intermediate	674.606	-8.23700	13.0	18.0
1	CHEMBL222813	CC(=O)N[C@H]1[C@H]([C@H](O)[C@H](O)CO)OC(C(=O)...	680.0	active	332.313	-3.57583	8.0	7.0
2	CHEMBL455019	CCC(CC)C(NC(C)=O)C1C(NC(=N)N)CC(C(=O)O)C1O	140.0	active	328.413	-0.13943	6.0	4.0
3	CHEMBL1643	NC(=O)c1ncn([C@@H]2O[C@H](CO)[C@@H](O)[C@H]2O)n1	20000.0	inactive	244.207	-3.01150	4.0	8.0
4	CHEMBL674	CCC(CC)O[C@@H]1C=C(C(=O)O)C[C@H](N)[C@H]1NC(C)=O	0.6	active	284.356	0.80690	3.0	4.0
...	...	...	...	...	...	...	...	...
93	CHEMBL4483331	C=C(C(=O)OC)[C@@H]1CC[C@H](C)[C@H]2C(=C(C)C(=O...	174600.0	inactive	319.401	2.17190	1.0	4.0
94	CHEMBL4554496	CC/C=C1/C=CC(=O)[C@@H](O)[C@]12C(=O)N(OC)[C@@]...	4100.0	intermediate	311.334	-0.06390	2.0	6.0
95	CHEMBL4578422	CC1(C)CC[C@]2(C(=O)N[C@@H](CCCNC(=N)N)C(=O)O)C...	14050.0	inactive	612.900	5.98167	6.0	4.0
96	CHEMBL1200340	CCOC(=O)C1=C[C@@H](OC(CC)CC)[C@H](NC(C)=O)[C@@...	100000.0	inactive	312.410	1.28540	2.0	5.0
97	CHEMBL4639026	CC(=O)c1cc(C)c(O)c(Cc2[nH]c3ccccc3c2Cc2c(O)c(C...	13.0	active	473.525	5.19394	5.0	6.0

In order to standardise the IC50 values and ease our interprtation we convert IC50 values to pIC50 , which is -log10(IC50 in molar units)

	molecule_chembl_id	canonical_smiles	class	MW	LogP	NumHDonors	NumHAcceptors	pIC50
0	CHEMBL502613	CC(=O)N[C@H]1[C@H]([C@H](O)[C@H](O)CO)O[C@](OC...	intermediate	674.606	-8.23700	13.0	18.0	5.522879
1	CHEMBL222813	CC(=O)N[C@H]1[C@H]([C@H](O)[C@H](O)CO)OC(C(=O)...	active	332.313	-3.57583	8.0	7.0	6.167491
2	CHEMBL455019	CCC(CC)C(NC(C)=O)C1C(NC(=N)N)CC(C(=O)O)C1O	active	328.413	-0.13943	6.0	4.0	6.853872
3	CHEMBL1643	NC(=O)c1ncn([C@@H]2O[C@H](CO)[C@@H](O)[C@H]2O)n1	inactive	244.207	-3.01150	4.0	8.0	4.698970
4	CHEMBL674	CCC(CC)O[C@@H]1C=C(C(=O)O)C[C@H](N)[C@H]1NC(C)=O	active	284.356	0.80690	3.0	4.0	9.221849
...	...	...	...	...	...	...	...	...
93	CHEMBL4483331	C=C(C(=O)OC)[C@@H]1CC[C@H](C)[C@H]2C(=C(C)C(=O...	inactive	319.401	2.17190	1.0	4.0	3.757956
94	CHEMBL4554496	CC/C=C1/C=CC(=O)[C@@H](O)[C@]12C(=O)N(OC)[C@@]...	intermediate	311.334	-0.06390	2.0	6.0	5.387216
95	CHEMBL4578422	CC1(C)CC[C@]2(C(=O)N[C@@H](CCCNC(=N)N)C(=O)O)C...	inactive	612.900	5.98167	6.0	4.0	4.852334

- We make use of PaDEL descriptors which are a set of molecular descriptors and fingerprints .There are a total of 881 descriptors. We use the IWSSr algorithm as a feature selection technique.

- In IWSSr, the SU criterion is used for weighting features.

$$SU_{i,c}(F_i, C) = 2\frac{H(F_i) - H(F_i|C)}{H(F_i) - H(C)}$$

	MW	LogP	NumHDonors	PubchemFP2	PubchemFP3	PubchemFP6	PubchemFP11	PubchemFP12	PubchemFP13	PubchemFP14	...	PubchemFP803	PubchemFP818	PubchemFP819
0	-0.262890	-1.911570	2.545291	0.532524	-0.459639	-0.29767	0.220863	-1.331989	-0.220863	0.622171	...	-0.220863	-0.532524	-0.478091
1	-0.288683	-0.643764	1.563699	0.532524	-0.459639	-0.29767	0.220863	-1.331989	-0.220863	0.622171	...	-0.220863	-0.532524	-0.478091
2	-0.845597	-1.703369	0.582107	-1.877849	-0.459639	-0.29767	0.220863	-1.331989	-0.220863	0.622171	...	-0.220863	-0.532524	-0.478091
3	-0.580063	-0.294631	0.091311	0.532524	-0.459639	-0.29767	0.220863	-1.331989	-0.220863	0.622171	...	-0.220863	-0.532524	-0.478091
4	-0.394522	-0.118095	-0.399485	0.532524	-0.459639	-0.29767	0.220863	0.750757	-0.220863	0.622171	...	-0.220863	-0.532524	-0.478091

- After performing Feature engineering , we choose the features with higher weights.

```
PubchemFP571    0.179283
PubchemFP643    0.179120
PubchemFP392    0.178218
PubchemFP528    0.177370
PubchemFP439    0.176637
                   ...
PubchemFP397    0.054887
PubchemFP414    0.054887
PubchemFP457    0.054887
PubchemFP459    0.054887
PubchemFP531    0.054887
Length: 300, dtype: float64
0.05488683227656127
```

```
{'PubchemFP0': 0.0,
 'PubchemFP1': 0.020815714202288265,
 'PubchemFP2': 0.13744651169866576,
 'PubchemFP3': 0.12463972561706874,
 'PubchemFP4': 0.0,
 'PubchemFP5': 0.0,
 'PubchemFP6': 0.08813660935834738,
 'PubchemFP7': 0.0,
 'PubchemFP8': 0.0,
 'PubchemFP9': 0.0,
 'PubchemFP10': 0.0,
 'PubchemFP11': 0.06003098472962928,
 'PubchemFP12': 0.163659836211245,
 'PubchemFP13': 0.06003098472962927,
 'PubchemFP14': 0.14755939032648574,
 'PubchemFP15': 0.16806585853031566,
 'PubchemFP16': 0.15250089092493746,
 'PubchemFP17': 0.0,
 'PubchemFP18': 0.09607685107815594,
 'PubchemFP19': 0.0984861781585627,
 'PubchemFP20': 0.1367001127435506,
 'PubchemFP21': 0.07449221373219574,
 'PubchemFP22': 0.03588960728519103,
 'PubchemFP23': 0.05107483732906649,
 'PubchemFP24': 0.03858760042190676,
...
 'PubchemFP876': 0.0,
 'PubchemFP877': 0.0,
 'PubchemFP878': 0.0,
 'PubchemFP879': 0.0,
 'PubchemFP880': 0.0}
```

- In our dataset we concluded that out of 881 features , ony 300 were relevant .

APPLYING THE MODELS

- KNN
- Deep Neural Network
- Random Forest Regressor

These are the three models which gave us best the results in terms of R2 score and MSE(Mean Squared Error)

Out of the above, the best ones turned out to be Deep Neural Networks and the K Nearest Neighbours.

K NEAREST NEIGHBOURS (KNN)

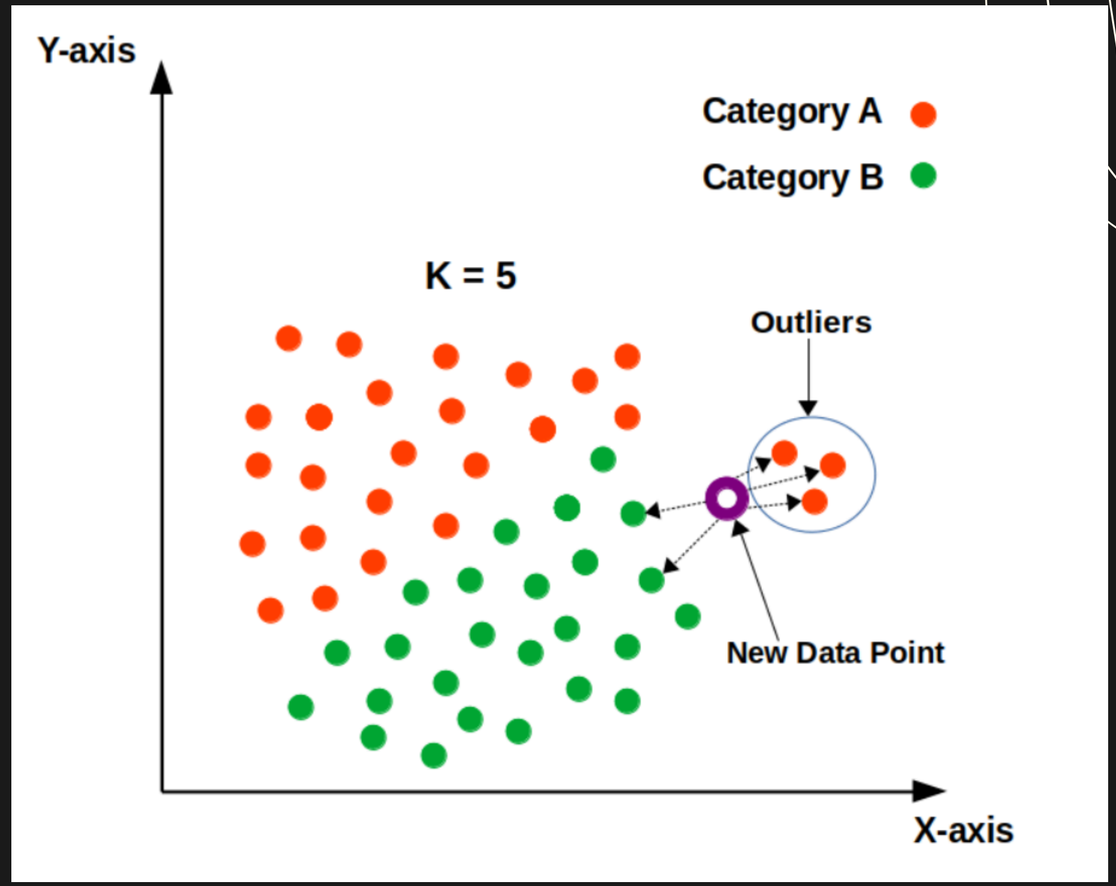
Motivation

- Easy to Implement: Simple and accessible for starting with more complex models.
- Non-Parametric: Does not assume any specific data distribution, useful for biological data.
- Local Patterns: Effectively captures local patterns, aiding in predicting protein-target interactions based on motifs.

Method

- Use GridSearchCV to find the optimal k value for KNN.
- Evaluate model performance using R^2 score and Mean Squared Error (MSE).

Coronavirus: R2 = 0.70, MSE = 0.32
H3N2: R2 = 0.47, MSE = 0.52



CODE SNAP

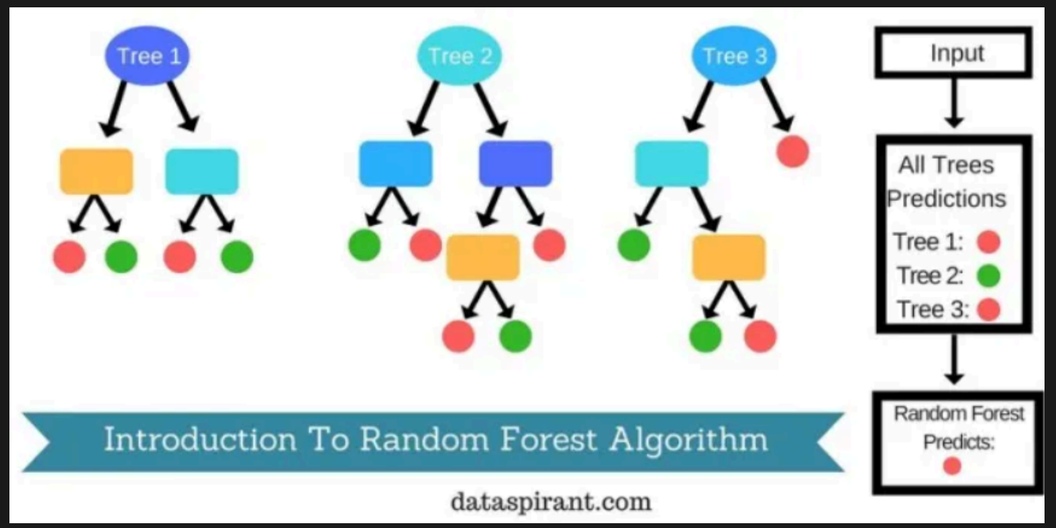
```python
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_neighbors': np.arange(20),
    'weights': ['uniform', 'distance'],
    'p': [1, 2],
    'metric': ['minkowski', 'euclidean', 'manhattan', 'chebyshev'],
}

from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

knn = KNeighborsRegressor()
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='r2', n_jobs=-1)
grid_search.fit(X_train, Y_train)
print(f'Best parameters: {grid_search.best_params_}')
print(f'Best R² score: {grid_search.best_score_:.2f}')
```

```
Best parameters: {'metric': 'minkowski', 'n_neighbors': 4, 'p': 1, 'weights': 'uniform'}
```

RANDOM FOREST REGRESSOR



Motivation

- Handling High-Dimensional Data:
- Feature Selection: Random Forest inherently performs feature selection, which is crucial when dealing with high-dimensional data common in protein-target interaction studies.
- Irrelevant Features: It can handle irrelevant or redundant features well, as it only uses a subset of features for each split
- Ensemble Learning: By averaging the predictions of multiple decision trees, Random Forest reduces the risk of underfitting, which is a common problem in complex biological datasets.
- Accuracy: Random Forest has been shown to provide high predictive accuracy in many bioinformatics applications, including protein-target interaction prediction.

Coronavirus: R2 = 0.64, MSE = 0.31

H3N2: R2 = 0.45, MSE = 0.6

CODE SNAP

```
model = RandomForestRegressor(n_estimators=100)
model.fit(X_train, Y_train)
Y_pred = model.predict(X_train)
r2 = model.score(X_test, Y_test)
r2
```

```
384157451992808
```

```
Y_pred = model.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error


mse = mean_squared_error(Y_test, Y_pred)
print(f'Mean Squared Error: {mse}')
```

```
n Squared Error: 0.3113585899120791
```

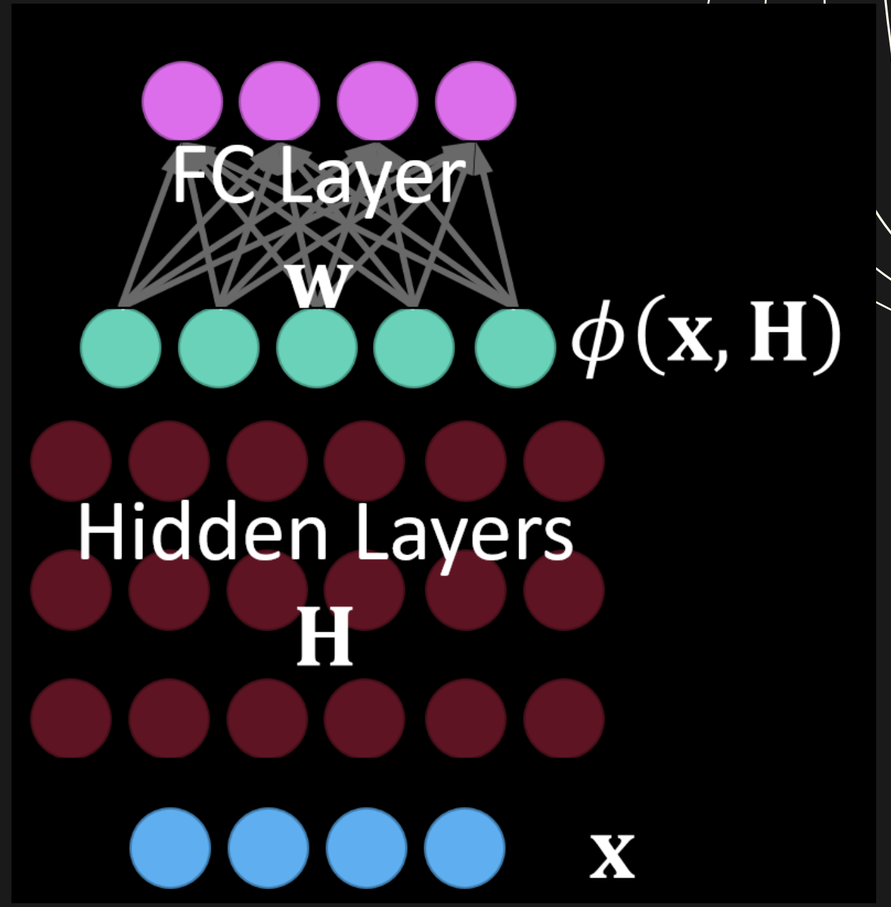
DEEP NEURAL NETWORK

Motivation

- Complex Data Handling: Neural networks manage high-dimensional data and intricate patterns effectively.
- Superior Performance: Often outperform traditional methods in prediction tasks due to their pattern-capturing ability.
- Handling Non-Linearity: Model non-linear relationships in protein-target interactions through layered structures and non-linear activation functions.

Method

- Remove Low Variance Features
- Experiment with Network Configurations: Layers, neurons, activation functions, optimization algorithms
- Select Best Parameters: Based on R^2 and MSE scores



Coronavirus: R2 = 0.70, MSE = 0.34
H3N2: R2 = 054, MSE = 0.7

CODE SNAP

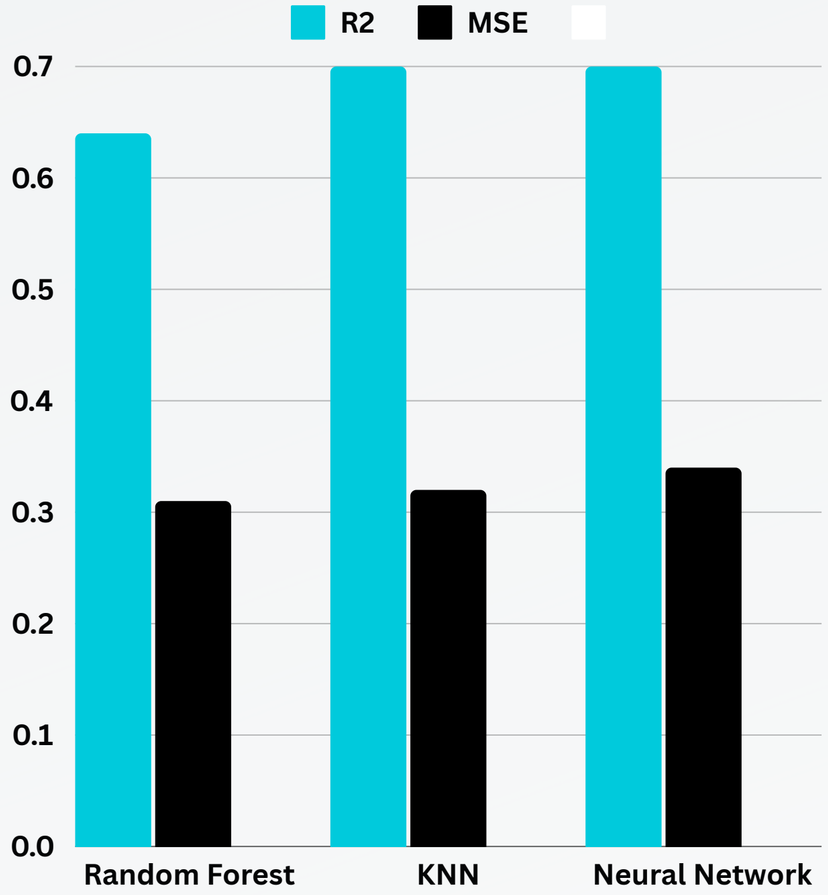
```
model=Sequential([
    Dense(1000, activation = 'relu', input_dim =472),
    Dense(500, activation='relu'),
    Dense(250, activation='relu'),
    Dense(100, activation='relu'),
    Dense(50, activation='relu'),
    Dense(1, activation='linear'),])
optimizer = Adam(learning_rate=0.0001)
model.compile(optimizer=optimizer,loss=MeanSquaredError())
```

```
history = model.fit(X_train, Y_train, epochs = 100, validation_split = 0.2)
```

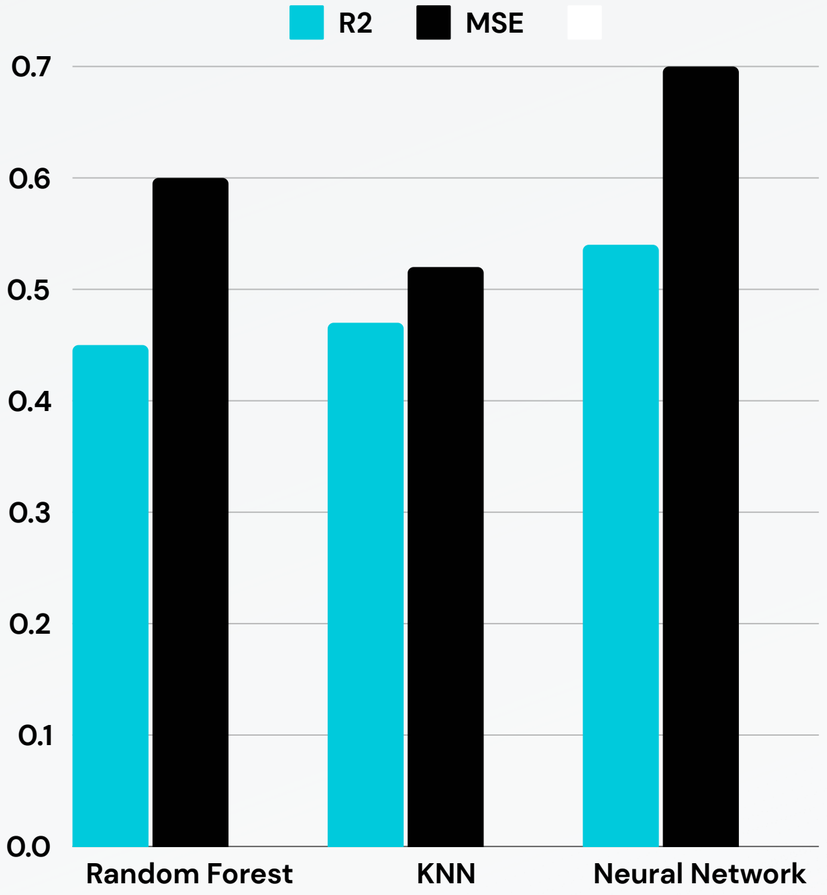
```
Y_pred = model.predict(X_test)
print("R2 Score:",r2_score(Y_test, Y_pred))
print("MSE Score:",mean_squared_error(Y_test, Y_pred))
```

```
1/1 [==============================] - 0s 89ms/step
R2 Score: 0.6956149773259229
MSE Score: 0.34490212693384126
```

RESULTS AND CONCLUSIONS



Evaluation Metrics For CORONA-Virus



Evaluation Metrics For H3N2

RESULTS AND CONCLUSIONS

- Clearly there is a difference in the evaluation metrics for the two datasets but even with the H3N2 dataset the least accuracy - 0.43 is acceptable enough for an organism.
- A primary reason for the difference in R2 scores is that the H3N2 dataset is for an entire organism and the coronavirus dataset is only for the membrane protein.
- The KNN model performs the best on the coronavirus proteinase dataset.
- Feature engineering and EDA greatly impact the performance. IWSSR got rid of 436 features in the engineered coronavirus dataset

A	B	C
	R2	MSE
Random Forest	0.64	0.31
KNN	0.7	0.32
Neural Network	0.7	0.34

A	B	C
	R2	MSE
Random Forest	0.45	0.6
KNN	0.47	0.52
Neural Network	0.54	0.7

THANK YOU