IN-SILICO COMPUTATION OF DRUG-TARGET BINDING AFFINITY PREDICTION FOR CANCER

A PROJECT REPORT

Submitted by

DIYA ARSHIYA S (2021115033)
PREETHAM VIJAYANDHRAN (2021115077)
SAMIUKKTHA DHARMALINGAM (2021115089)
SATHYADHARINI SRINIVASAN (2021115097)

A report for the phase-I of the project
submitted to the Faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment
for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600 025

OCTOBER 2025

ANNA UNIVERSITY

CHENNAI - 600 025

BONA FIDE CERTIFICATE

Certified that this project report titled In-Silico Computation of Drug-Target Binding Affinity Prediction for Cancer is the bona fide work of Diya Arshiya S (2021115033), Preetham Vijayandhran (2021115077), Samiukkktha Dharmalingam (2021115089), Sathyadharini Srinivasan (2021115097) who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

PLACE: Chennai
DATE: 17.10.2024

Dr. Mala T
DEPARTMENT OF IST, CEG
ANNA UNIVERSITY
CHENNAI 600025

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600025

ABSTRACT

Accurate prediction of drug-target binding affinity (DTA) is essential for accelerating drug discovery and improving the identification of potential therapeutic candidates. This study explores various computational approaches to DTA prediction, each utilizing different strategies for modeling the interactions between drugs and proteins. By analyzing molecular features, interaction patterns, and critical subsequences within drug and protein representations, these methods aim to enhance prediction accuracy and efficiency.

The performance of each approach is evaluated based on key metrics, including accuracy, interpretability, and computational efficiency. The insights gained from this evaluation will guide the development of a quantum-based DTA prediction model. Quantum computing's potential to model complex molecular interactions is anticipated to provide significant advancements in predictive accuracy, scalability, and the ability to interpret drug-target relationships, thereby contributing to the future of drug discovery.

ACKNOWLEDGEMENT

It is our privilege to express our deepest sense of gratitude and sincere thanks to Dr. Mala T, Professor, Project Guide, Department of Information Science and Technology, College of Engineering, Guindy, Anna University, for her constant supervision, encouragement, and support in our project work. We greatly appreciate the constructive advice and motivation that was given to help us advance our project in the right direction.

We are grateful to Dr. S. Swamynathan, Professor and Head, Department of Information Science and Technology, College of Engineering Guindy, Anna University for providing us with the opportunity and necessary resources to do this project.

We would also wish to express our deepest sense of gratitude to the Members of the Project Review Committee: Dr. J. Indumathi, Professor, Dr. S. Sendhil Kumar, Professor, Dr. P Geetha , Associate Professor, Dr. D. Narashiman, Teaching Fellow Department of Information Science and Technology ,College of Engineering Guindy, Anna University, for their guidance and useful suggestions that were beneficial in helping us improve our project.

We also thank the faculty members and non teaching staff members of the Department of Information Science and Technology, Anna University, Chennai for their valuable support throughout the course of our project work.

Diya Arshiya S (2021115033)
Preetham Vijayandhran (2021115077)
Samiukktha Dharmalingam (2021115089)
Sathyadharini Srinivasan(2021115097)

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Accurate prediction of drug-target binding affinity (DTA) plays a pivotal role in drug discovery and development, enabling researchers to rapidly identify promising therapeutic candidates. To advance this objective, we analyze three prominent computational models—SimBoost, Affinity2Vec, and AttentionDTA—that approach DTA prediction through distinct methodologies. By examining the capabilities and limitations of these models, we aim to build a solid foundation for developing a quantum-based DTA prediction model that leverages quantum computing's unique potential for modeling complex molecular interactions.

SimBoost, a similarity-based boosting model, utilizes pairwise similarities between drugs and targets to estimate binding affinities, incorporating various molecular and biological features that strengthen its predictive accuracy. Affinity2Vec adopts an embedding-based strategy, generating vector representations of drugs and proteins that capture intricate patterns in their interactions, allowing the model to generalize effectively to new drug-protein pairs. Meanwhile, AttentionDTA, a deep learning framework, processes drugs and proteins through SMILES representations and amino acid sequences, using an attention mechanism to identify critical molecular subsequences impacting binding affinity.

Each model's performance was rigorously evaluated using established metrics, encompassing accuracy, interpretability, and computational efficiency. This analysis provides a comprehensive assessment of each model's strengths and weaknesses in predicting drug-target interactions. Insights gained from these evaluations will guide the design of a quantum model, seeking to overcome existing limitations and harness quantum computing to enhance accuracy, interpretability, and scalability in DTA prediction.

CHAPTER 2
LITERATURE SURVEY

Title of the Article	**Journals/Conference Details and Year of Publication**	**Highlights of Approaches Followed**	**Challenges to be Addressed**
Prediction of Drug-Target Affinity Using Attention Neural Network	Tang, X.; Lei, X.; Zhang, Y. *Int. J. Mol. Sci.* 2024	GRA-DTA integrates GraphSAGE, BiGRU, and ANN for improved DTA prediction. Achieves lower MSE and higher CI on KIBA and Davis datasets.	High computational demands due to complex model architecture. Dependence on the quality and availability of DTA datasets.
Drug Target Interaction Prediction using Graph Convolution based Neural Fingerprinting	A. Joshy, G. C. Kasyap, P. D. Reddy, I. T. Anjusha and K. A. Abdul Nazeer, *2022 IEEE 19th India Council International Conference (INDICON)*, Kochi, India	NFPCNN integrates Neural Fingerprinting and 1D CNNs for accurate DTI prediction, focusing on drug repurposing for orphan diseases. Demonstrates strong performance on the KIBA dataset with low	Balancing computational efficiency with model complexity in NFPCNN. Ensuring data quality and diversity for reliable DTI predictions.

		MSE, outperforming models like SimBoost and DeepDTA.	

SimBoost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines	He, T., Heidemeyer, M., Ban, F. *et al* *J Cheminform* 9, 24 (2017)	SimBoost improves continuous DTA prediction over binary classification, offering nuanced insights. Outperforms KronRLS and Matrix Factorization on datasets like Davis, Metz, and KIBA.	Handling missing data and distinguishing it from true negative interactions. Managing the computational complexity of feature construction and GBM training
CutQC: using small Quantum computers for large Quantum circuit evaluations	Wei Tang, Teague Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. Association for Computing Machinery, New York, NY, USA	CutQC enables evaluation of circuits up to 100 qubits. Extends quantum device capabilities using hybrid computing.	Tackling noise and low qubit counts in NISQ devices. Managing hybrid quantum-classical integration complexity.

Quantum Machine Learning: A tutorial	José D. Martín-Guerrero, Lucas Lamata. Neurocomputing,Volume 470,2022,Pages 457-461,ISSN 0925-2312	QML offers speedups by enhancing ML with quantum resources. Provides a clear classification of QML approaches and applications.	Sensitivity to parameters like Gaussian kernel length scale. Bridging theory and practical implementation in QML.

CHAPTER 3
IMPLEMENTATION

3.1 FEATURE EXTRACTION

The KIBA dataset used in this project contains information about drugs and their corresponding target proteins. The drug data is represented in SMILES (Simplified Molecular Input Line Entry System) format, while the target proteins are provided as amino acid sequences.

3.1.1 Extracting Relevant Features for Drugs

- For drugs, molecular descriptors are extracted using the RDKit library. Specifically, the SMILES strings are converted into molecular representations that allow the computation of various molecular-level descriptors. This is done through the following RDKit commands and code snippet:

```
from rdkit import Chem
from rdkit.Chem import Descriptors

        # Function -> extract_features(smiles_input,0)
mol = Chem.MolFromSmiles(smiles)
st.write(f"\n**Extracting features for SMILES:** `{smiles}`")
st.write(f"Molecular Weight: {Descriptors.MolWt(mol)}")
st.write(f"logP: {Descriptors.MolLogP(mol)}")
st.write(f"TPSA: {Descriptors.TPSA(mol)}")
atom_types = []
atom_degrees = []
for atom in mol.GetAtoms():
    atom_types.append(atom.GetAtomicNum())
    atom_degrees.append(atom.GetDegree())
```

- rdkit.Chem helps parse the SMILES strings, and Descriptors allows us to extract properties such as molecular weight, LogP, the number of hydrogen bond acceptors and atom specific properties as well.

3.1.2 Extracting Relevant Features for Proteins

- For proteins, the target sequence is analyzed using BioPython's ProteinAnalysis tool, which converts the sequence into meaningful descriptors, such as molecular weight, amino acid composition, and isoelectric point. This is achieved using:

```
from Bio.SeqUtils.ProtParam import ProteinAnalysis

#Function -> extract_protein_descriptors(protein_sequence_input,op)
protein = ProteinAnalysis(protein_sequence)
molecular_weight = protein.molecular_weight()
aromaticity = protein.aromaticity()

residue_counts = protein.count_amino_acids()
aliphaticity = sum([residue_counts.get(aa, 0) for aa in aliphatic_residues])

hydrophobicity_scale_pH2 = {'A': 1.8, 'V': 4.2, 'I': 4.5, 'L': 3.8}
isoelectric_point = protein.isoelectric_point()
hydrophobicity_pH2 = sum([hydrophobicity_scale_pH2.get(aa, 0) for aa in protein_sequence]) / len(protein_sequence)
```

- With this, descriptors related to protein structure and characteristics are extracted.

3.1.3 Normalization and Feature Preparation

Once the molecular descriptors for both drugs and proteins have been extracted, the features are normalized to ensure uniformity across the dataset. Since different drugs and proteins may have varying lengths, padding is applied to make the features compatible for further processing. After this step, the features are concatenated, combining the drug and protein features into a single array for each pair. These combined feature vectors are then saved into a dataset for further analysis.

```
protein_list=extract_protein_descriptors(protein_sequence_input,0)
drug_list=extract_features(smiles_input,0)
combined_features = drug_list+protein_list #Concatenation
```

3.2 MODEL IMPLEMENTATIONS

For this project, we have implemented three different models to evaluate the performance and effectiveness of various approaches in solving the problem at hand. The models are designed based on different techniques, each offering unique insights into the data and contributing to a comprehensive analysis. Below, we provide a detailed explanation of each model.

3.2.1 SimBoost

SimBoost is a machine learning model specifically designed for drug-target binding affinity (DTA) prediction. It is a similarity based boosting model that uses pairwise similarities between drug-drug and target-target along with other features to predict the binding affinity score.

3.2.1.1 SimBoost Implementation

A. Inputs

- The **drug-target binding affinity matrix** gives the affinity score between each drug-target pair.
- The **drug-drug similarity matrix** shows how similar two drugs are based on their chemical structures (e.g., using Tanimoto similarity).
- The **target-target similarity matrix** shows the similarity between two targets, possibly based on sequence identity or structural information.

B. Feature Extraction for Drugs and Targets

From these matrices, we extract seven features for each drug and target to enhance the prediction process.

- **Average Similarity**: This feature is calculated by averaging the similarity values from the target-target similarity matrix for each target. It captures how similar each target is to other targets in the dataset. The average similarity for

each drug is calculated similarly.

```
for i in sim_targets:
    target_gene_names.loc[i,'t_avg-sim']=np.mean(sim_targets.loc[i,:])
```

- **Average Binding Affinity**: This feature is derived by averaging the binding affinity values from the drug-target binding affinity matrix across all drugs. It reflects the overall binding behavior of the target with the drug set. The average binding affinity for each drug is calculated similarly.

```
target_bindingvals={}
for i in range(len(drug_target_binding)):
    x=drug_target_binding.iloc[i,:] #drug=x[0],target=x[1],binding_val=[2]
    if x[1] not in target_bindingvals:
        target_bindingvals[x[1]]=[x[2]]
    else:
        target_bindingvals[x[1]].append(x[2])
for i in target_bindingvals:
    target_gene_names.loc[i,'t_avg-binding']=np.mean(target_bindingvals[i])
```

- **Graph Creation**: A graph is created where each target is a node, and the edges between nodes are formed if similarity scores between targets is above a certain threshold value. The threshold value taken is the mean of average similarity for each target calculated as the first feature.
- **Number of Neighbors in the Graph**: For each target node, this feature represents the number of direct connections (neighbors) it has in the similarity graph. A higher number of neighbors suggests that the target shares structural or functional properties with other targets. The number of neighbors for each drug is calculated similarly.

```
for vertex in target_graph.vs:
target_gene_names.loc[sorted(id1)[vertex.index], 't_n_neighbors'] =
target_graph.neighborhood_size(vertex,mindist=1)
```

- **PageRank from the Graph**: Using the PageRank algorithm on the target similarity graph, we compute the PageRank score for each target. This score indicates the importance of a target based on its connections within the network. Targets with higher PageRank have stronger or more numerous connections, suggesting a central role in the similarity network. The page rank for each drug is calculated similarly.

  ```
  target_gene_names.loc[sorted(id1)[vertex.index], 't_page_rank'] =
  target_graph.pagerank(vertex)
  ```

- **Non-Negative Matrix Factorization (NMF) for Latent Features**: Non-negative matrix factorization is applied to the target-target similarity matrix to extract three latent features. These features capture underlying patterns in the similarity data, potentially representing hidden biological or structural properties. NMF aims to factorize the binding affinity matrix $B \in \mathbb{R}_{+}^{d \times t}$, where 'd' represents the number of drugs and 't' represents the number of targets, into two non-negative matrices, $P \in \mathbb{R}_{+}^{k \times d}$, a matrix representing the latent features for the drugs, where k is the number of latent factors or components. $Q \in \mathbb{R}_{+}^{k \times t}$, a matrix representing the latent features of the targets.

C. Model Training with Gradient Boosting

- The core model used in SimBoost is based on Gradient Boosting Machines (GBMs), where decision trees are built sequentially, and the model learns from the errors of previous trees.Libraries such as XGBoost or LightGBM can be used for the gradient boosting algorithm.

D. Evaluate the Model

- After training, evaluate the model on the test dataset using performance metrics such as **Root Mean Square Error (RMSE)**, accuracy, precision, recall, F1score.

3.2.2 AttentionDTA

AttentionDTA is a deep learning model designed to predict the binding affinity between drugs and target proteins. It uses a sequence-based approach, focusing on the SMILES representation for drugs and amino acid sequences for proteins.Its core innovation is the attention mechanism that allows the model to identify key subsequences in both the drug and protein that are significant for predicting binding affinity.Providing insights into which regions of the drug and protein contribute to the predicted affinity.

3.2.2.1 AttentionDTA implementation

A. Inputs:

- **Drug Sequence (SMILES)**: Linear representation of a drug's chemical structure.
- **Protein Sequence**: String of amino acids representing the protein.

B. Feature Extraction:

1. **Tokenization**:

- Convert **SMILES** and **Protein sequence** into numerical representations(numerical encoding).

```
def tokenize(sequence):
    return [vocab[ch] for ch in sequence]  # Tokenize based on vocabulary
```

2. Embedding:

- Use embedding layers to transform tokenized sequences into **low-dimensional vectors**.

```
drug_embedding = nn.Embedding(vocab_size, embed_size)
```

```
protein_embedding = nn.Embedding(vocab_size, embed_size)
```

C. Attention Mechanism:

Apply **self-attention** to focus on relevant parts of drug and protein sequences. Calculate attention using:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

```
class Attention(nn.Module):

 def __init__(self, d_model, num_heads):

super(Attention, self).__init__()

self.attention = nn.MultiheadAttention(embed_dim=d_model, num_heads=num_heads)
```

3. Concatenation:

- Merge the **attention-refined features** into a single vector.

```
def concatenate_features(drug_feat, protein_feat):

return torch.cat((drug_feat, protein_feat), dim=-1)
```

D. Model Training:

1. **Fully Connected Neural Network (FCNN)**: Pass **concatenated features** through fully connected layers to predict binding affinity.

```
x = F.relu(fc1(concatenated_features))

affinity_score = output_layer(x)
```

2. **Loss Function:** Use **Mean Squared Error (MSE)** to minimize prediction errors.

```
criterion = nn.MSELoss()
```

```
loss = criterion(predicted_affinity, true_affinity)
```

E. Model Evaluation: Evaluate using metrics such as **MSE**, **MAE**, and **R2**.

```
mse = mean_squared_error(y_true, y_pred)

mae = mean_absolute_error(y_true, y_pred)

r2 = r2_score(y_true, y_pred)
```

3.2.2.2 Key Components:

- **Attention Layer**: Captures relevant interactions.
- **Embeddings**: Converts sequences into meaningful vectors.
- **Concatenation Layer**: Merges features.
- **Fully Connected Layers**: Outputs binding affinity score.

Libraries Used:

- **TensorFlow/Keras**: For neural network development.
- **Scikit-learn**: For performance evaluation.

This implementation efficiently integrates attention mechanisms to learn drug-target interactions.

3.2.3 AFFINITY2VEC:

Affinity2Vec is a method for learning vector embeddings that represent the interaction between drugs and proteins. It leverages drug and protein embeddings to predict drug-target affinities, with applications in drug discovery and repositioning. Through this embedding-based approach, the model captures the underlying patterns in drug-target interactions, enabling more accurate predictions of potential interactions in unseen data.

1. Dataset Preparation

Drug and Target IDs:
The model reads **drug IDs from a file** and protein gene names for the dataset.

```
# Load drug IDs
drug_ids = pd.read_csv('drug_PubChem_CIDs.txt', header=None)[0].tolist()
# Load target gene names
target_ids = pd.read_csv('target_gene_names.txt', header=None)[0].tolist()
```

Interaction Matrix:
Drug-target interaction affinities are loaded into a matrix.

```
# Load the drug-target interaction affinities
aff = pd.read_csv('drug-target_interaction_affinities_Kd__Davis_et_al.2011v1.txt', sep='\t')
DrPr_Matrix = aff.values  # Convert to matrix
```

2. Loading Similarity Matrices

Drug-Drug Similarities: A matrix of drug-drug similarities is loaded.

```
# Load drug-drug similarities
Dr_SimM = np.loadtxt('drug_drug_similarity_matrix.txt')
```

Target-Target Similarities: Target-target similarities are loaded.

```
# Load target-target similarities
Pr_SimM = np.loadtxt('target_target_similarity_matrix.txt')
```

3. Normalization of Similarities: Both similarity matrices are **normalized**.

```
# Normalize the similarity matrices
Dr_SimM = normalizedMatrix(Dr_SimM)
Pr_SimM = normalizedMatrix(Pr_SimM)
```

4. Creating Pairwise Drug-Target Labels: Pairs of each drug with each target are created, storing the interaction values.

```
pairX = []
label = []

for i, j in itertools.product(range(aff.shape[0]), range(aff.shape[1])):
    pairX.append((drug_ids[i], target_ids[j]))
    label.append(aff.iloc[i, j])  # Append corresponding affinity values
```

5. Transforming Affinity Scores: Affinity values are transformed for better interpretability.

```
# Transform affinity values using logarithm
aff_mat = np.log1p(np.array(label))
YY = np.array(aff_mat)
```

6. Embedding Generation:

Seq2Seq Drug Embeddings: Drug representations are generated.

```
# Generate drug embeddings
drEMBED = generateDrugEmbeddings(drug_ids)
```

ProtVec Protein Embeddings: Protein target representations are generated.

```
# Generate protein embeddings
prEMBED = generateProteinEmbeddings(target_ids)
```

7. Embedding-Based Similarity Calculation: Cosine similarity for embeddings is calculated.

```
# Calculate cosine similarity for drug embeddings
Dr_SimM = cosine_similarity(drEMBED)
# Normalize drug similarities
Dr_SimM = normalizedMatrix(Dr_SimM)

# Calculate cosine similarity for protein embeddings
Pr_SimM = cosine_similarity(prEMBED)
# Normalize protein similarities
Pr_SimM = normalizedMatrix(Pr_SimM)
```

8. Creating Feature Vectors: For each drug-target pair, embeddings are concatenated.

```
concatenateFV = []
for i, j in itertools.product(range(aff.shape[0]), range(aff.shape[1])):
    features = list(drEMBED[i]) + list(prEMBED[j])
    concatenateFV.append(features)

ConcatFV_seq = np.array(concatenateFV)
print('ConcatFV shape', ConcatFV_seq.shape)
```

9. Meta-Path Features: Meta-path features are computed from similarities.

```
# Example code to compute meta-path features
meta_path_features = computeMetaPathFeatures(Dr_SimM, Pr_SimM)
```

10. Training and Testing Split: The dataset is divided into training and test sets.

```
# Load train and test split
train_indices = np.loadtxt('train_fold_setting1.txt', dtype=int)
test_indices = np.loadtxt('test_fold_setting1.txt', dtype=int)
```

11. Machine Learning Regressor

Define the regression model.

```
from xgboost import XGBRegressor

xg_reg = XGBRegressor(learning_rate=0.1, max_depth=5, n_estimators=100)
```

12. Model Training

The model is fitted on the training set and performance is evaluated.

```
# Train and evaluate the model for each fold
for train_index in train_indices:
    X_train = ConcatFV_seq[train_index]
    Y_train = YY[train_index]
    xg_reg.fit(X_train, Y_train)
    # Validation and testing can be done similarly
```

13. Final Evaluation

Performance metrics are evaluated.

```
#  Evaluating metrics
from sklearn.metrics import mean_squared_error, r2_score

Y_pred = xg_reg.predict(X_test)
mse = mean_squared_error(Y_test, Y_pred)
ci = concordance_index(Y_test, Y_pred)  # Custom implementation needed
```

14. Visualization

Predicted vs. actual affinity values are plotted.

```
import matplotlib.pyplot as plt

plt.scatter(Y_test, Y_pred)
plt.xlabel('Actual Affinity')
plt.ylabel('Predicted Affinity')
plt.title('Predicted vs Actual Affinity')
plt.show()
```

This structured explanation provides a clear overview of the Affinity2Vec model workflow, detailing the methodology and implementation steps effectively.

CHAPTER 4
CONCLUSION AND FUTURE WORKS

In this project, we implemented and evaluated three distinct models aimed at predicting drug-target binding affinity. Each model provided valuable insights into the prediction task, with varying levels of accuracy and precision. Below is a summary of their performance:

4.1 SimBoost

Accuracy=0.957; Precision=0.824; RMSE=0.501

4.2 Affinity2Vec

Average_AUPR=0.923 , RMSE=1.50, Concordance=0.62

4.3 AttentionDTA

MSE =0.3323 (0.0119); MAE =0.3905 (0.0201); R2 =0.5255 (0.0183); RMSE=0.5764

To achieve novelty and increase the accuracy and time complexity, we will focus on developing a **q**uantum-based model in Phase 2. This model will leverage the computational power of quantum mechanics to address issues such as:

- Scalability: Quantum algorithms can efficiently process larger datasets, which is a limitation for classical models.
- Feature Complexity: Quantum computing can handle the complexity of high-dimensional feature spaces, making it ideal for capturing intricate relationships in the data.
- Optimization: Quantum-based methods are expected to improve accuracy and precision by optimizing model parameters in ways that classical algorithms struggle to achieve.

By integrating a quantum-based approach, we aim to push the boundaries of predictive modeling and enhance the overall performance, especially in areas like [specifics of the problem, e.g., drug-target affinity prediction].