First Phrase

Get library

```
# Normal Lib
import os
import pandas as pd
import numpy as np
import pickle
import math
```

```
# keras
from keras import Model
import keras
from keras.layers import Conv3D, Input, MaxPooling3D, BatchNormalization, Dense, Dropout, Flatten, AveragePooling
import tensorflow as tf
from keras.optimizers import Adam, RMSprop, SGD
from keras.regularizers import L2

# Metrics
from scipy.stats import pearsonr # Pearson R best
from keras.metrics import AUC, MeanAbsoluteError, Precision, Recall, Accuracy
from sklearn.metrics import matthews_corrcoef
from keras.activations import linear

# Import File
from zipfile import ZipFile
from google.colab import drive
```

```
!pip install rdkit
from rdkit import Chem
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting rdkit
  Downloading rdkit-2023.3.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (29.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 29.7/29.7 MB 43.7 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from rdkit) (1.22.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from rdkit) (8.4.0)
Installing collected packages: rdkit
Successfully installed rdkit-2023.3.1
```

```
drive.mount('/content/gdrive/')
dataset_folder = '/content/gdrive/MyDrive/Final'
files = os.listdir(dataset_folder)
for file in files:
  if '.zip' in file:
    file_path = os.path.join(dataset_folder, file)
    with ZipFile(file_path, 'r') as f:
      f.extractall()
```

```
Mounted at /content/gdrive/
```

Get data and labels zip

```
protein_folder = '/content/protein'
protein_list = os.listdir(protein_folder)

ligand_folder = '/content/ligand'
ligand_list = os.listdir(ligand_folder)

label_folder = '/content/label'
label_list = os.listdir(label_folder)
```

```
totalSize = len(ligand_list)
```

```
totalSize
```

```
16686
```

```
# permu = np.random.permutation(totalSize)
permu = np.random.RandomState(seed=69).permutation(totalSize)


train_num, validate_num, test_num = 0,0,0


iDataset_num = totalSize
ratio = (60,20,20)


train_num = int(iDataset_num * (ratio[0]/ (ratio[0]+ratio[1]+ratio[2])))
# val_num = int(iDataset_num * (ratio[1]/ (ratio[0]+ratio[1]+ratio[2])))
# test_num = int(iDataset_num * (ratio[2]/ (ratio[0]+ratio[1]+ratio[2])))

val_num = 100
test_num = 500
last_num = 2000


train_list_IDs = permu[:train_num]
val_list_IDs = permu[train_num:(train_num+val_num)]
test_list_IDs = permu[(train_num+val_num):(train_num+val_num+test_num)]
last_list_IDs = permu[(train_num+val_num+test_num):(train_num+val_num+test_num+last_num)]


train_list_IDs
```

```
array([ 2788, 12876,  5452, ..., 13517, 15820, 11375])
```

Get features

```
def get_atom_features(atom, amino_acid, isprotein):
    ATOM_CODES = {}
    metals = ([3, 4, 11, 12, 13] + list(range(19, 32))
              + list(range(37, 51)) + list(range(55, 84))
              + list(range(87, 104)))
    atom_classes = [(5, 'B'), (6, 'C'), (7, 'N'), (8, 'O'), (15, 'P'), (16, 'S'), (34, 'Se'),
                    ([9, 17, 35, 53], 'halogen'), (metals, 'metal')]
    for code, (atomidx, name) in enumerate(atom_classes):
        if type(atomidx) is list:
            for a in atomidx:
                ATOM_CODES[a] = code
        else:
            ATOM_CODES[atomidx] = code
    try:
        classes = ATOM_CODES[atom.GetAtomicNum()]
    except:
        classes = 9

    possible_chirality_list = [
        Chem.rdchem.ChiralType.CHI_UNSPECIFIED,
        Chem.rdchem.ChiralType.CHI_TETRAHEDRAL_CW,
        Chem.rdchem.ChiralType.CHI_TETRAHEDRAL_CCW,
        Chem.rdchem.ChiralType.CHI_OTHER
    ]
    chirality = possible_chirality_list.index(atom.GetChiralTag())

    possible_formal_charge_list = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
    try:
        charge = possible_formal_charge_list.index(atom.GetFormalCharge())
    except:
        charge = 11

    possible_hybridization_list = [
        Chem.rdchem.HybridizationType.S,
        Chem.rdchem.HybridizationType.SP,
        Chem.rdchem.HybridizationType.SP2,
```

```
        Chem.rdchem.HybridizationType.SP3,
        Chem.rdchem.HybridizationType.SP3D,
        Chem.rdchem.HybridizationType.SP3D2,
        Chem.rdchem.HybridizationType.UNSPECIFIED
    ]
    try:
        hyb = possible_hybridization_list.index(atom.GetHybridization())
    except:
        hyb = 6

    possible_numH_list = [0, 1, 2, 3, 4, 5, 6, 7, 8]
    try:
        numH = possible_numH_list.index(atom.GetTotalNumHs())
    except:
        numH = 9

    possible_implicit_valence_list = [0, 1, 2, 3, 4, 5, 6, 7]
    try:
        valence = possible_implicit_valence_list.index(atom.GetTotalValence())
    except:
        valence = 8

    possible_degree_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    try:
        degree = possible_degree_list.index(atom.GetTotalDegree())
    except:
        degree = 11

    is_aromatic = [False, True]
    aromatic = is_aromatic.index(atom.GetIsAromatic())

    mass = atom.GetMass() / 100

    # idx = atom.GetIdx()
    # with open(protein_path, 'r+') as f:
    #     readlines = f.readlines()
    #     f.close()

    amino_acids = [
        'ALA', 'ARG', 'ASN', 'ASN', 'ASP', 'CYS', 'GLU', 'GLN', 'GLY', 'HIS', 'ILE', 'LEU', 'LYS', 'MET', 'PHE',
    ]
    if amino_acid in amino_acids:
      amino_acid = amino_acids.index(amino_acid)
    else:
      amino_acid = int(len(amino_acids) + 1)

    # amino_acid = amino_acids.index(amino_acid)
    # amino_acid = 0
    # for lines in readlines:
    #     if 'HETATM' in lines or 'ATOM' in lines:
    #         if idx == int(lines[6:11]):
    #             amino_acid = lines[17:20]
                # if amino_acid in amino_acids:
                #     amino_acid = amino_acids.index(amino_acid)
                # else:
                #     amino_acid = int(len(amino_acids) + 1)

    return [classes, chirality, charge, hyb, numH, valence, degree, aromatic, mass, amino_acid, isprotein]
```

Get min coordinates

```
def get_min(compound_positions):
    minx,miny,minz = 999,999,999
    for pos in compound_positions:
        x, y, z = pos
        if x < minx:
            minx = x

        if y < miny:
```

```
            miny = y

        if z < minz:
            minz = z

    return (minx,miny,minz)
```

Get grid

```
def adjust_grid(compound, compound_positions, protein_path, isprotein, grid, minx, miny, minz):
    atoms_aa = []
    with open(protein_path, 'r+') as f:
        readlines = f.readlines()
        f.close()

    for idx, lines in enumerate(readlines):
        if 'HETATM' in lines or 'ATOM' in lines:
            # print(idx)
            # if (idx+1) == int(lines[6:11]):
            atoms_aa.append(lines[17:20])

                # if amino_acid in amino_acids:
                #     amino_acid = amino_acids.index(amino_acid)
                # else:
                #     amino_acid = int(len(amino_acids) + 1)

    for idx, pos in enumerate(compound_positions):
        x,y,z = pos

        amino_acid = atoms_aa[idx]
        atom = compound.GetAtomWithIdx(int(idx))
        features = get_atom_features(atom, amino_acid, isprotein)
        features.extend(pos)

        if grid[round(x - minx), round(y - miny), round(z - minz)][0] == 0:
            grid[round(x - minx), round(y - miny), round(z - minz)] = features

        elif grid[round(x - minx)+1, round(y - miny), round(z - minz)][0] == 0:
            grid[round(x - minx)+1, round(y - miny), round(z - minz)] = features

        elif grid[round(x - minx), round(y - miny)+1, round(z - minz)][0] == 0:
            grid[round(x - minx), round(y - miny)+1, round(z - minz)] = features

        elif grid[round(x - minx), round(y - miny), round(z - minz)+1][0] == 0:
            grid[round(x - minx), round(y - miny), round(z - minz)+1] = features

        elif grid[round(x - minx)+1, round(y - miny)+1, round(z - minz)][0] == 0:
            grid[round(x - minx)+1, round(y - miny)+1, round(z - minz)] = features

        elif grid[round(x - minx), round(y - miny)+1, round(z - minz)+1][0] == 0:
            grid[round(x - minx), round(y - miny)+1, round(z - minz)+1] = features

        elif grid[round(x - minx)+1, round(y - miny), round(z - minz)+1][0] == 0:
            grid[round(x - minx)+1, round(y - miny), round(z - minz)+1] = features

        elif grid[round(x - minx)+1, round(y - miny)+1, round(z - minz)+1][0] == 0:
            grid[round(x - minx)+1, round(y - miny)+1, round(z - minz)+1] = features

    return grid


def set_grid(protein_path, isprotein= 1):
    compound = Chem.MolFromPDBFile(protein_path, False, False, 1)
    compound_conf = compound.GetConformer()
    compound_positions = compound_conf.GetPositions()

    atom = compound.GetAtomWithIdx(int(1))
    features = get_atom_features(atom, '', isprotein)

    minx, miny, minz = get_min(compound_positions)
```

```
    grid=np.zeros((52,52,52,len(features)+3))

    grid = adjust_grid(compound, compound_positions, protein_path, isprotein, grid, minx, miny, minz)

    return grid, minx, miny, minz
```

Add ligand

```
def add_ligand(ligand_path, grid, minx, miny, minz, isprotein = 0):
    ligand = Chem.MolFromPDBFile(ligand_path, False, False, 1)
    ligand_conf = ligand.GetConformer()
    ligand_positions = ligand_conf.GetPositions()

    grid = adjust_grid(ligand, ligand_positions, ligand_path, isprotein, grid, minx, miny, minz)

    return grid
```

Create Protein_Grid_list

```
def create_grid_list(protein_folder):
    protein_list = os.listdir(protein_folder)
    grids = []
    for protein in protein_list:
        protein_name = protein.split('.')[0]
        protein_path = os.path.join(protein_folder, protein)

        grid, minx, miny, minz = set_grid(protein_path)
        grids.append((protein_name, grid, (minx, miny, minz)))
    return grids
```

Get grid list

```
grids = create_grid_list(protein_folder)

grid = grids[0][1]
minx, miny, minz = grids[0][2]
name = grids[0][0]

new_grid = add_ligand('/content/ligand/5c1u-FIP_model79.pdb', grid, minx, miny, minz)

# for grid in grids:
count = 0
for x in new_grid:
    for y in x:
      for features in y:
        if features[0] != 0:
          # print(features)
          count+=1
print(count)
```

Get batch data

```
# /content/ligand/3qzq-10b_model1.pdb

def get_data_batch(dataset_idx, protein_folder, ligand_folder, ligand_list, label_folder, label_list, batch_size,
  # dataset = get_dataset_ID(dataset_idx) # Wrong af
  baList = []
  statList = []
  gridList = []

  batch_list = [value for idx, value in enumerate(dataset_idx) if idx >= index * batch_size and idx < (index+1)*b
```

```
    # print(batch_list)
    for i in batch_list:

      complexFile = ligand_list[i]

      # Get Data and Labels
      from_protein = complexFile.split('-')[0]
      complex_name = complexFile.split('.')[0]
      protein_path = os.path.join(protein_folder, from_protein+'.pdb')
      complexFile_path = os.path.join(ligand_folder, complexFile)

      grid, minx, miny, minz = set_grid(protein_path)
      grid = add_ligand(complexFile_path, grid, minx, miny, minz)
      gridList.append(grid)
      grid = []

      protein = [value for value in label_list if from_protein in value][0]
      label_file_path = os.path.join(label_folder, protein)

      df = pd.read_csv(label_file_path)
      listidx = df.index[df['file.pdb'] == complex_name].tolist()[0]
      ba = df['BA'][listidx]
      stat = df['Hit/No_hit'][listidx]
      if stat == 'hit':
        stat = 1
      else:
        stat = 0
      baList.append(ba)
      statList.append(stat)

    gridList = np.array(gridList)
    baList = np.array(baList)
    statList = np.array(statList)

    return gridList, baList, statList
```

Get metric

```
def get_metrics(y_label, y_pred, ytype):

    if ytype == 0: # Regression
      print("++++++++++++++++++++++++++Regression++++++++++++++++++++++++++")
      PearsonR, _ = pearsonr(y_label, y_pred)
      print('Pearson Correlation Coefficient: ' + str(PearsonR))
      MSE = MeanAbsoluteError()
      MSE.update_state(y_label, y_pred)
      MSE = MSE.result().numpy()
      print('Mean Absolute Error: ' + str(MSE))
      # RMSE = math.sqrt(MSE)
      # print('Root Mean Absolute Error: ' + str(RMSE))

      return PearsonR, MSE

    if ytype == 1: # Classification
      print("++++++++++++++++++++++Classification++++++++++++++++++++++")
      auc = keras.metrics.AUC()
      tp = keras.metrics.TruePositives(thresholds= 0.9)
      tn = keras.metrics.TrueNegatives(thresholds= 0.9)
      fp = keras.metrics.FalsePositives(thresholds= 0.9)
      fn = keras.metrics.FalseNegatives(thresholds= 0.9)

      # auc.update_state(y_label, y_pred)
      tp.update_state(y_label, y_pred)
      tn.update_state(y_label, y_pred)
      fp.update_state(y_label, y_pred)
      fn.update_state(y_label, y_pred)

      # auc = auc.result().numpy()
```

```
    tp = tp.result().numpy()
    tn = tn.result().numpy()
    fp = fp.result().numpy()
    fn = fn.result().numpy()

    precision = tp/ (tp+fp) # PPV
    print('Precision: ' + str(precision))

    # recall = Recall()
    # recall.update_state(y_label, y_pred)
    # recall = recall.result().numpy()
    recall = tp/(tp+fn) # Recall - TPR
    print('Recall: ' + str(recall))

    specificity = tn/(tn+fp)
    print('Specificity: ' + str(specificity))

    NPV = tn/(tn+fn)
    print('NPV: ' + str(NPV))

    # accuracy = Accuracy()
    # accuracy.update_state(y_label, y_pred)
    # accuracy = accuracy.result().numpy()
    # print('AUC: ' + str(auc))

    # f1_score = 2 * (precision * recall) / (precision + recall)
    # print('F1_Score: ' + str(f1_score))

    MCC = (tp*tn - fp*fn)/ math.sqrt(   (tp+fp)*(tp+fn)*(tn+fp)*(tn+fn)  ) # Phi coefficient
    print("Phi coefficient:" + str(MCC))

    return precision, recall, specificity, NPV, MCC
```

Get Validate

```
import csv

def model_val_dataset(val_dataset_idx, protein_folder, label_folder, label_list, batch_size, epochs, save_path, b
  dataset_len = len(val_dataset_idx)
  runs = dataset_len // batch_size
  last_batch = dataset_len - batch_size*runs
  model = keras.models.load_model(save_path)
  # csv_path = 'resultCSV.csv'
  # csv_path = os.path.join(best_path, csv_path)
  # csvfile = open(csv_path, 'w')
  # fields = ['Model', 'Prediction C', 'Label C', 'Prediction R', 'Prediction R']
  # writer = csv.writer(f)
  # writer = csv.writer(csvfile)
  # writer.writerow(fields)
  # PearsonR_list, MCC_list, RSME_list = 0,0,0
  ba_Actual, stat_Actual, ba_Pred, stat_Pred = [], [], [], []
  print("----------------------- Start ValDataset -----------------------")
  for i in range(int(runs+1)):

    print("Get dataset on batch "+str(i+1))
    if i != runs+1:
      gridList, baList, statList = get_data_batch(val_dataset_idx, protein_folder, ligand_folder, ligand_list, la
    else:
      gridList, baList, statList = get_data_batch(val_dataset_idx, protein_folder, ligand_folder, ligand_list, la

    print("----------------------- Predict ValDataset -----------------------")
    result = model.predict(gridList, verbose=2)
    gridList = []
    pred_reg, pred_class = result

    baList = [value for value in baList.tolist()]
    statList = [value for value in statList.tolist()]
```

```
    pred_reg = [value[0] for value in pred_reg.tolist()]
    pred_class = [value[0] for value in pred_class.tolist()]

    if ba_Actual == []:
      ba_Actual = baList
      stat_Actual = statList
      ba_Pred = pred_reg
      stat_Pred = pred_class
    else:
      ba_Actual.extend(baList)
      stat_Actual.extend(statList)
      ba_Pred.extend(pred_reg)
      stat_Pred.extend(pred_class)

    baList, statList = [], []

  PearsonR, MSE = get_metrics(ba_Actual, ba_Pred, 0)
  print("-------------------------------------------------------------")
  precision, recall, specificity, NPV, MCC = get_metrics(stat_Actual, stat_Pred, 1)
  # for idx, trueIndex in enumerate(val_dataset_idx):
  #   row = [str(trueIndex), str(stat_Actual[idx]), str(stat_Pred[idx]), str(ba_Actual[idx]), str(ba_Pred[idx])]
  #   print(row)
  #   writer.writerow(row)
  # csvfile.close()
  return PearsonR, MSE, precision, recall, specificity, NPV, MCC, ba_Actual, ba_Pred, stat_Actual, stat_Pred
```

Get Train

```
def model_train_dataset(model, train_dataset_idx, val_dataset_idx, protein_folder, label_folder, label_list, batc
  dataset_len = len(train_dataset_idx)
  runs = dataset_len // batch_size
  cur = 1
  # PearsonR, MSE, RMSE, precision, recall, auc, f1_score, MCC = 0,0,0,0,0,0,0,0
  # checkPearsonR, checkMCC, checkRMSE = 0,0,999
  log_txt = "log.txt"
  log_path = os.path.join(save_path, log_txt)
  readline = ''
  if os.path.exists(log_path):
    log_file = open(log_path,"r+")
    readline = log_file.readline()
    log_file.close()
  else:
    with open(log_path, 'w+') as f:
      f.write('0/'+str(runs))
      f.close()

  if readline == '' or int(readline.split('/')[0]) > runs or int(readline.split('/')[0]) == 0:
    cur = 1
    model.save(save_path)
  else:
    cur = int(readline.split('/')[0])

  check = 0

  print("---------------------- Start TrainDataset ------------------------")
  for i in range(int(cur-1),int(runs+1)):
    print("======================Batch "+ str(i+1)+"==============================")
    model = keras.models.load_model(save_path)
    print("Get dataset")
    # get_data_batch(dataset_idx, protein_folder, ligand_folder, ligand_list, label_folder, label_list, batch_siz
    # get_data_batch(train_list_IDs, protein_folder, ligand_folder, ligand_list, label_folder, label_list, 32, 0)
    if i != runs+1:
      gridList, baList, statList = get_data_batch(val_dataset_idx, protein_folder, ligand_folder, ligand_list, la
    else:
      gridList, baList, statList = get_data_batch(val_dataset_idx, protein_folder, ligand_folder, ligand_list, la

    print("---------------------- Train TrainDataset ------------------------")
    model.fit(gridList, [baList, statList], epochs= epochs,verbose=0)
    gridList, baList, statList = [], [], []
    # PearsonR, MSE, RMSE, precision, recall, auc, f1_score, MCC = model_val_dataset(val_dataset_idx, protein_fol
```

```
      print("Save")
      model.save(save_path)
      log_file = open(log_path,"r+")
      readline = log_file.write(str(i)+'/'+str(runs))
      log_file.close()
      # if PearsonR > checkPearsonR and MCC > checkMCC and RMSE < checkRMSE:
      #   model.save(best_path)
      #   checkPearsonR = PearsonR
      #   checkMCC = MCC
      #   checkRMSE = RMSE
      if check == 0 and batch_size*i >= 2000:
        check +=1
        model.save(best_path)
    return model
```

Get Model

```
def combine_embedding(drop_rate, input_shape= (52,52,52,11)):
  inp = Input(shape= input_shape, name='Input_Complexes')

  ## Check there are atoms
  x1 = Conv3D(filters= 8, kernel_size=(1,1,1), padding='same', bias_initializer='zeros', kernel_initializer='glor
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)

  x1 = Conv3D(8, kernel_size=(3,3,3))(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)

  x1 = Conv3D(32,kernel_size=(3,3,3),padding='same')(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)
  x1 = MaxPooling3D(pool_size=2)(x1)

  x1 = Conv3D(64,kernel_size=(1,1,1),padding='same')(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)

  x1 = Conv3D(64,kernel_size=(3,3,3),padding='same')(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)

  x1 = MaxPooling3D(pool_size=2)(x1)

  x1 = Conv3D(128,kernel_size=(1,1,1),padding='same')(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)

  x1 = Conv3D(128,kernel_size=(3,3,3),padding='same')(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)
  x1 = MaxPooling3D(pool_size=2)(x1)

  x1 = Conv3D(256,kernel_size=(1,1,1),padding='same')(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)

  x1 = Conv3D(256,kernel_size=(3,3,3),padding='same')(x1)
  x1 = BatchNormalization()(x1)
  x1 = Activation('relu')(x1)
  x1 = MaxPooling3D(pool_size=2)(x1)

    # Global Pooling
  x2 = GlobalAveragePooling3D()(x1)

  x2 = Dense(256)(x2)
  x2 = BatchNormalization()(x2)
  x2 = Activation('relu')(x2)
  x2 = Dropout(0.5)(x2)
```

```
    # Flattening
    x1 = Flatten()(x1)

    x1 = Dense(256)(x1)
    x1 = BatchNormalization()(x1)
    x1 = Activation('relu')(x1)
    x1 = Dropout(0.5)(x1)

    # Regression Output
    d1 = Dense(1,kernel_regularizer=tf.keras.regularizers.l2(0.01))(x1)
    # Classification Output
    d2 = Dense(1, activation='sigmoid')(x2)

    return Model(inputs=[inp], outputs=[d1,d2], name='Embedding')
```

Create model

```
fake_grid, x,y,z = set_grid('/content/protein/3qzq.pdb')
```

```
np.shape(fake_grid)
```

```
(52, 52, 52, 14)
```

```
shape = np.shape(fake_grid)
shape
```

```
(52, 52, 52, 14)
```

```
model = combine_embedding(0.5, shape)
model.summary()
```

```
Model: "Embedding"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 Input_Complexes (InputLayer)   [(None, 52, 52, 52,  0           []
                                 14)]

 conv3d (Conv3D)                (None, 52, 52, 52,   120         ['Input_Complexes[0][0]']
                                8)

 batch_normalization (BatchNorm  (None, 52, 52, 52,   32         ['conv3d[0][0]']
 alization)                     8)

 activation (Activation)        (None, 52, 52, 52,   0           ['batch_normalization[0][0]']
                                8)

 conv3d_1 (Conv3D)              (None, 50, 50, 50,   1736        ['activation[0][0]']
                                8)

 batch_normalization_1 (BatchNo  (None, 50, 50, 50,   32         ['conv3d_1[0][0]']
 rmalization)                   8)

 activation_1 (Activation)      (None, 50, 50, 50,   0           ['batch_normalization_1[0][0]']
                                8)

 conv3d_2 (Conv3D)              (None, 50, 50, 50,   6944        ['activation_1[0][0]']
                                32)

 batch_normalization_2 (BatchNo  (None, 50, 50, 50,   128        ['conv3d_2[0][0]']
 rmalization)                   32)

 activation_2 (Activation)      (None, 50, 50, 50,   0           ['batch_normalization_2[0][0]']
                                32)

 max_pooling3d (MaxPooling3D)   (None, 25, 25, 25,   0           ['activation_2[0][0]']
                                32)

 conv3d_3 (Conv3D)              (None, 25, 25, 25,   2112        ['max_pooling3d[0][0]']
                                64)

 batch_normalization_3 (BatchNo  (None, 25, 25, 25,   256        ['conv3d_3[0][0]']
 rmalization)                   64)

 activation_3 (Activation)      (None, 25, 25, 25,   0           ['batch_normalization_3[0][0]']
                                64)

 conv3d_4 (Conv3D)              (None, 25, 25, 25,   110656      ['activation_3[0][0]']
```

```
                                64)

    batch_normalization_4 (BatchNo  (None, 25, 25, 25,   256        ['conv3d_4[0][0]']
    rmalization)                    64)

    activation_4 (Activation)       (None, 25, 25, 25,   0          ['batch_normalization_4[0][0]']
                                    64)

    max_pooling3d_1 (MaxPooling3D)  (None, 12, 12, 12,   0          ['activation_4[0][0]']
                                    64)
```

```
optimizer=Adam(learning_rate=1e-4)

loss=['mean_squared_error', "binary_crossentropy"]
model.compile(optimizer= optimizer,
              loss= loss, run_eagerly=True)
```

Save paths

```
# save_path = '/content/gdrive/MyDrive/Final/Model'
# best_path = '/content/gdrive/MyDrive/Final/Best'

save_path = '/content/gdrive/MyDrive/SFCNN/Model1'
best_path = '/content/gdrive/MyDrive/SFCNN/Best1'
```

Set hyperparameters

```
batch_size = 32
epochs = 150
```

New test

```
model = keras.models.load_model('/content/gdrive/MyDrive/Final/Model')
```

```
model.summary()
```

```
Model: "Embedding"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 Input_Complexes (InputLayer)   [(None, 52, 52, 52,  0           []
                                 14)]

 conv3d (Conv3D)                (None, 52, 52, 52,   24256       ['Input_Complexes[0][0]']
                                64)

 batch_normalization (BatchNorm  (None, 52, 52, 52,   256        ['conv3d[0][0]']
 alization)                     64)

 activation (Activation)        (None, 52, 52, 52,   0           ['batch_normalization[0][0]']
                                64)

 max_pooling3d (MaxPooling3D)   (None, 26, 26, 26,   0           ['activation[0][0]']
                                64)

 conv3d_1 (Conv3D)              (None, 26, 26, 26,   221312      ['max_pooling3d[0][0]']
                                128)

 batch_normalization_1 (BatchNo  (None, 26, 26, 26,   512        ['conv3d_1[0][0]']
 rmalization)                   128)

 activation_1 (Activation)      (None, 26, 26, 26,   0           ['batch_normalization_1[0][0]']
                                128)

 average_pooling3d (AveragePool  (None, 13, 13, 13,   0          ['activation_1[0][0]']
 ing3D)                         128)

 conv3d_2 (Conv3D)              (None, 13, 13, 13,   884992      ['average_pooling3d[0][0]']
                                256)

 activation_2 (Activation)      (None, 13, 13, 13,   0           ['conv3d_2[0][0]']
                                256)
```

```
batch_normalization_2 (BatchNo  (None, 13, 13, 13,   1024       ['activation_2[0][0]']
rmalization)                    256)

average_pooling3d_1 (AveragePo  (None, 7, 7, 7, 256  0          ['batch_normalization_2[0][0]']
oling3D)                        )

conv3d_3 (Conv3D)               (None, 7, 7, 7, 256  1769728     ['average_pooling3d_1[0][0]']
                                )

batch_normalization_3 (BatchNo  (None, 7, 7, 7, 256  1024       ['conv3d_3[0][0]']
rmalization)                    )

activation_3 (Activation)       (None, 7, 7, 7, 256  0           ['batch_normalization_3[0][0]']
                                )

average_pooling3d_2 (AveragePo  (None, 4, 4, 4, 256  0          ['activation_3[0][0]']
oling3D)                        )

flatten (Flatten)               (None, 16384)        0           ['average_pooling3d_2[0][0]']
```

```
# PearsonR, MSE, precision, recall, specificity, NPV, MCC = model_val_dataset(test_list_IDs, protein_folder, labe
```

```
# !pip install mayavi
```

```
# test1_list_IDs = permu[12000:12100]
```

```
# len(test1_list_IDs)
```

```
PearsonR, MSE, precision, recall, specificity, NPV, MCC, ba_Actual, ba_Pred, stat_Actual, stat_Pred = model_val_d
```

```
PearsonR, MSE, precision, recall, specificity, NPV, MCC, ba_Actual, ba_Pred, stat_Actual, stat_Pred = model_val_d
```

```
---------------------- Start ValDataset -----------------------
Get dataset on batch 1
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 360ms/epoch - 360ms/step
Get dataset on batch 2
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 230ms/epoch - 230ms/step
Get dataset on batch 3
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 228ms/epoch - 228ms/step
Get dataset on batch 4
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 228ms/epoch - 228ms/step
Get dataset on batch 5
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 231ms/epoch - 231ms/step
Get dataset on batch 6
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 228ms/epoch - 228ms/step
Get dataset on batch 7
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 249ms/epoch - 249ms/step
Get dataset on batch 8
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 230ms/epoch - 230ms/step
Get dataset on batch 9
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 230ms/epoch - 230ms/step
Get dataset on batch 10
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 232ms/epoch - 232ms/step
Get dataset on batch 11
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 231ms/epoch - 231ms/step
Get dataset on batch 12
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 231ms/epoch - 231ms/step
Get dataset on batch 13
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 227ms/epoch - 227ms/step
Get dataset on batch 14
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 227ms/epoch - 227ms/step
Get dataset on batch 15
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 229ms/epoch - 229ms/step
Get dataset on batch 16
---------------------- Predict ValDataset -----------------------
1/1 - 1s - 1s/epoch - 1s/step
++++++++++++++++++++++++++Regression++++++++++++++++++++++++++
Pearson Correlation Coefficient: 0.37141046168240693
```

```
Mean Absolute Error: 0.44962853
------------------------------------------------------------
++++++++++++++++++++++Classification++++++++++++++++++++++
Precision: 0.9872881
Recall: 0.97899157
Specificity: 0.9885496
NPV: 0.9810606
```

```
PearsonR, MSE, precision, recall, specificity, NPV, MCC, ba_Actual, ba_Pred, stat_Actual, stat_Pred = model_val_c
```

```
---------------------- Start ValDataset -----------------------
Get dataset on batch 1
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 480ms/epoch - 480ms/step
Get dataset on batch 2
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 228ms/epoch - 228ms/step
Get dataset on batch 3
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 229ms/epoch - 229ms/step
Get dataset on batch 4
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 235ms/epoch - 235ms/step
Get dataset on batch 5
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 230ms/epoch - 230ms/step
Get dataset on batch 6
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 232ms/epoch - 232ms/step
Get dataset on batch 7
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 229ms/epoch - 229ms/step
Get dataset on batch 8
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 235ms/epoch - 235ms/step
Get dataset on batch 9
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 231ms/epoch - 231ms/step
Get dataset on batch 10
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 227ms/epoch - 227ms/step
Get dataset on batch 11
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 227ms/epoch - 227ms/step
Get dataset on batch 12
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 229ms/epoch - 229ms/step
Get dataset on batch 13
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 226ms/epoch - 226ms/step
Get dataset on batch 14
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 227ms/epoch - 227ms/step
Get dataset on batch 15
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 225ms/epoch - 225ms/step
Get dataset on batch 16
---------------------- Predict ValDataset -----------------------
1/1 - 0s - 363ms/epoch - 363ms/step
++++++++++++++++++++++++++Regression++++++++++++++++++++++++++
Pearson Correlation Coefficient: 0.37141046168240693
Mean Absolute Error: 0.44962853
------------------------------------------------------------
++++++++++++++++++++++Classification++++++++++++++++++++++
Precision: 0.9872881
Recall: 0.97899157
Specificity: 0.9885496
NPV: 0.9810606
```

```
# len(stat_Pred)
```

```
fields = ['Model', 'Hit_Label', 'Hit_Prediction', 'BindingAffinity_Label', 'BindingAffinity_Prediction']
```

```
rows = []
```

```
for idx, value in enumerate(val_list_IDs):
  row = []
  row.append(ligand_list[value])
  row.append(stat_Actual[idx])
  row.append(stat_Pred[idx])
  row.append(ba_Actual[idx])
  row.append(ba_Pred[idx])
  # row = (ligand_list[value], stat_Actual[idx], stat_Pred[idx], ba_Actual[idx], ba_Pred[idx])
```

```
  # print(type(row))
  rows.append(row)
  # if rows == []:
  #   rows = row
  # else:
  #   rows.append(row)
```

```
rows
```

```
[['5c1u-Q_model15.pdb', 1, 0.9993299245834351, -6.4, -6.485359191894531],
 ['3sjo-FOPMC_model46.pdb', 1, 0.9996739625930786, -7.4, -6.979996204376221],
 ['3qzr-FIOMC_model22.pdb',
  1,
  5.735444210586138e-05,
  -6.8,
  -6.977699279785156],
 ['3qzq-R_model41.pdb', 1, 0.9994035959243774, -7.5, -8.023859977722168],
 ['3sjo-NK18k_model4.pdb', 1, 0.9993780851364136, -7.1, -6.983725070953369],
 ['5gso-SG_model5.pdb', 1, 0.9997405409812927, -6.7, -7.3869123458862305],
 ['7dnc-10b_model81.pdb', 1, 0.9991466999053955, -7.1, -7.336747646331787],
 ['5gsw-D_model54.pdb', 1, 0.999691367149353, -6.5, -6.8474907875061035],
 ['5gsw-CIP_model58.pdb', 1, 0.9996819496154785, -6.2, -6.279407501220703],
 ['3sjo-8v_model31.pdb', 0, 7.358544098678976e-05, -7.1, -7.324412822723389],
 ['3qzq-F_model34.pdb', 0, 0.006644252687692642, -6.2, -6.533586502075195],
 ['5c1u-NK19k_model23.pdb', 1, 0.9996753931045532, -7.2, -6.904038906097412],
 ['3qzq-R_model8.pdb', 1, 0.9988143444061279, -6.9, -7.713748931884766],
 ['3qzq-Q_model93.pdb', 0, 0.0064354585483670235, -6.2, -6.543240070343018],
 ['3sjo-CR_model82.pdb', 0, 0.000303973036352545, -6.6, -6.959396839141846],
 ['5c1u-10b_model59.pdb', 0, 0.001484525273554027, -6.9, -6.52389669418335],
 ['3sjo-AG_model64.pdb', 0, 7.263942734425655e-06, -6.4, -7.164269924163818],
 ['5gso-CIP_model32.pdb', 0, 0.00025730679044499993, -6.7, -7.318888187408447],
 ['7dnc-CIP_model93.pdb', 0, 8.597495252615772e-06, -6.4, -6.014207363128662],
 ['5gsw-CIP_model24.pdb', 1, 0.9996488094329834, -6.6, -6.323408126831055],
 ['5gsw-CIP_model59.pdb', 0, 0.06277648359537125, -6.2, -6.703273296356201],
 ['5gsw-AG_model88.pdb', 1, 0.9998031258583069, -7.1, -6.934863567352295],
 ['5gsw-8x_model11.pdb', 1, 0.9998950958251953, -7.2, -7.430840015411377],
 ['5gso-FIOMC_model81.pdb', 1, 0.9996964931488037, -8.4, -7.312397003173828],
 ['3sjk-NK18k_model70.pdb', 1, 0.9987233281135559, -6.9, -7.010770320892334],
 ['5gsw-SG_model31.pdb', 1, 0.9995075464248657, -6.9, -7.274509906768799],
 ['5gso-NK19k_model56.pdb', 1, 0.9996870756149292, -6.5, -7.398436069488525],
 ['7dnc-10b_model72.pdb', 1, 0.9993628859519958, -7.0, -7.303957462310791],
 ['7dnc-HF_model87.pdb', 1, 0.9993100166320801, -6.7, -7.24800968170166],
 ['3qzq-NK18k_model14.pdb',
  0,
  0.00011404198448872194,
  -6.4,
  -6.627558708190918],
 ['3sjo-D_model96.pdb', 0, 9.499493899056688e-05, -6.5, -6.8696393966674805],
 ['3sjk-CR_model100.pdb', 0, 5.487958696903661e-05, -6.0, -6.875138282775879],
 ['3sjk-D_model23.pdb', 0, 6.638530612690374e-05, -7.0, -6.836582183837891],
 ['5gso-D_model77.pdb', 1, 0.999344527721405, -6.3, -7.104061126708984],
 ['5c1u-10b_model25.pdb', 1, 0.999481737613678, -7.2, -6.862899303436279],
 ['5c1u-NK18k_model48.pdb', 1, 0.9977472424507141, -7.2, -6.761124134063721],
 ['5gso-NK18k_model70.pdb', 1, 0.9995132684707642, -7.1, -7.090506076812744],
 ['5gsw-FIP_model88.pdb', 0, 0.0736018493771553, -6.5, -6.601014137268066],
 ['5gso-CR_model85.pdb', 1, 0.9992552399635315, -6.4, -6.959052562713623],
 ['5gsw-F_model2.pdb', 1, 0.9996814727783203, -6.8, -6.5274529457092285],
 ['3qzr-8w_model49.pdb', 0, 0.0005750557756982744, -6.3, -6.898451328277588],
 ['7dnc-FOPMC_model90.pdb', 1, 0.9997428059577942, -7.6, -7.509217739105225],
 ['3qzq-SG_model51.pdb', 0, 0.00033046415774151683, -6.7, -6.713829040527344],
 ['3qzq-9_model54.pdb', 0, 0.00037561042699962854, -6.6, -6.924108028411865],
 ['5gso-L_model15.pdb', 1, 0.9993953704833984, -7.3, -7.098631858825684],
 ['3sjo-L_model100.pdb', 1, 0.9979116320610046, -7.0, -6.916303634643555],
 ['3sjk-NK18k_model86.pdb', 1, 0.9994366765022278, -7.4, -7.141674041748047],
 ['3qzq-FOPMC_model5.pdb', 1, 0.998946487903595, -6.5, -7.30050802230835],
 ['3qzq-10b_model38.pdb', 0, 0.0031556379981338978, -6.6, -6.517406463623047],
 ['5gso-FIP_model9.pdb', 0, 0.0017458174843341112, -6.4, -7.308454513549805],
```

```
len(rows)
```

```
0
```

```
csv_path = '/content/gdrive/MyDrive/Final/CSV/resultSFCNN.csv'
```

```
with open(csv_path, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(fields)

    # writing the data rows
    csvwriter.writerows(rows)


datatset_path = '/content/gdrive/MyDrive/Final/CSV/dataset.csv'


fields = ['Model', 'Protein', 'Ligand']


rows = []


count = 0
```

Draw

```
try:
  import py3Dmol
except:
  !pip install py3Dmol
  import py3Dmol
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting py3Dmol
  Downloading py3Dmol-2.0.3-py2.py3-none-any.whl (12 kB)
Installing collected packages: py3Dmol
Successfully installed py3Dmol-2.0.3
```

```
14748
```

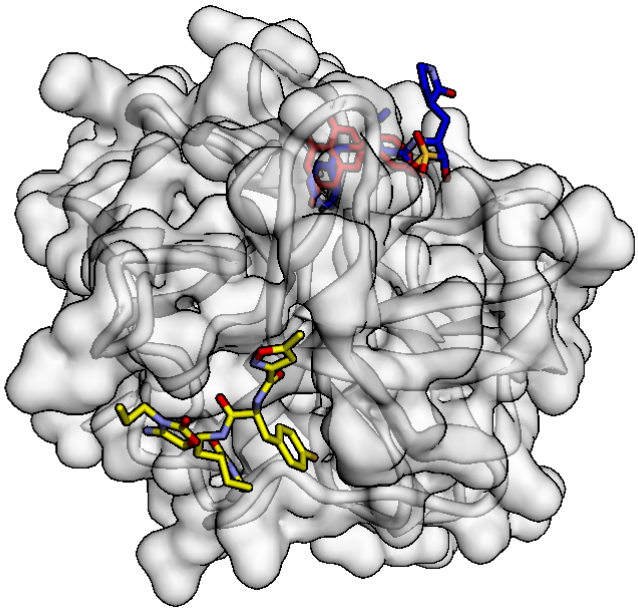
```
14748
```

```
ligand_14748 = ligand_list[14748]
ligand_14748
```

```
'3qzq-L_model16.pdb'
```

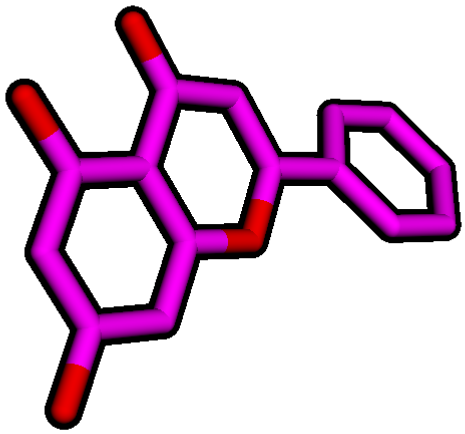
```
# ligand_14748_path = os.path.join(ligand_folder, ligand_14748)
hitLead_Ligand = '/content/ligand/5c1u-CR_model22.pdb'


hitLead_Protein = '/content/protein/5c1u.pdb'


# ligand_14748_path = os.path.join(ligand_folder, ligand_14748)
hit_Ligand = '/content/ligand/5c1u-GC376_model97.pdb'


hit_Protein = '/content/protein/3qzr.pdb'


no_Ligand = '/content/ligand/5c1u-FOPMC_model71.pdb'


no_Protein = '/content/protein/3qzr.pdb'


view = py3Dmol.view()
view.removeAllModels()
view.setViewStyle({'style':'outline','color':'black','width':0.1})

view.addModel(open(hitLead_Protein,'r').read(),format='pdb')
Prot=view.getModel()
Prot.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'white'}})
view.addSurface(py3Dmol.VDW,{'opacity':0.8,'color':'white'})
```

```
view.addModel(open(hitLead_Ligand,'r').read(),format='mol2')
ref_m1 = view.getModel()
ref_m1.setStyle({},{'stick':{'colorscheme':'redCarbon','radius':0.2}})

view.addModel(open(hit_Ligand,'r').read(),format='mol2')
ref_m2 = view.getModel()
ref_m2.setStyle({},{'stick':{'colorscheme':'blueCarbon','radius':0.2}})

view.addModel(open(no_Ligand,'r').read(),format='mol2')
ref_m3 = view.getModel()
ref_m3.setStyle({},{'stick':{'colorscheme':'yellowCarbon','radius':0.2}})

# ref_m.setStyle({},{'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})
# ref_m.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})


# results=Chem.SDMolSupplier('1AZ8_lig_vina_out.sdf')

# p=Chem.MolToMolBlock(results[0],False)

# print('Reference: Magenta | Vina Pose: Cyan')
# print ('Pose: {} | Score: {}'.format(results[0].GetProp('Pose'),results[0].GetProp('Score')))

# view.addModel(p,'mol')
# x = view.getModel()
# x.setStyle({},{'stick':{'colorscheme':'cyanCarbon','radius':0.2}})

view.zoomTo()
view.show()
```

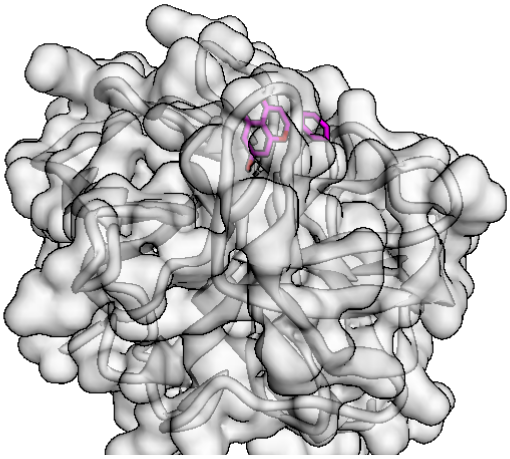


```
view = py3Dmol.view()
view.removeAllModels()
view.setViewStyle({'style':'outline','color':'black','width':0.1})

# view.addModel(open(hitLead_Protein,'r').read(),format='pdb')
# Prot=view.getModel()
# Prot.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'white'}})
# view.addSurface(py3Dmol.VDW,{'opacity':0.8,'color':'white'})


view.addModel(open(hitLead_Ligand,'r').read(),format='mol2')
ref_m = view.getModel()
ref_m.setStyle({},{'stick':{'colorscheme':'magentaCarbon','radius':0.2}})
# ref_m.setStyle({},{'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})
# ref_m.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})


# results=Chem.SDMolSupplier('1AZ8_lig_vina_out.sdf')
```

```
# p=Chem.MolToMolBlock(results[0],False)

# print('Reference: Magenta | Vina Pose: Cyan')
# print ('Pose: {} | Score: {}'.format(results[0].GetProp('Pose'),results[0].GetProp('Score')))

# view.addModel(p,'mol')
# x = view.getModel()
# x.setStyle({},{'stick':{'colorscheme':'cyanCarbon','radius':0.2}})

view.zoomTo()
view.show()
```



```
view = py3Dmol.view()
view.removeAllModels()
view.setViewStyle({'style':'outline','color':'black','width':0.1})

view.addModel(open(hitLead_Protein,'r').read(),format='pdb')
Prot=view.getModel()
Prot.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'white'}})
view.addSurface(py3Dmol.VDW,{'opacity':0.8,'color':'white'})


view.addModel(open(hitLead_Ligand,'r').read(),format='mol2')
ref_m = view.getModel()
ref_m.setStyle({},{'stick':{'colorscheme':'magentaCarbon','radius':0.2}})
# ref_m.setStyle({},{'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})
# ref_m.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})


# results=Chem.SDMolSupplier('1AZ8_lig_vina_out.sdf')

# p=Chem.MolToMolBlock(results[0],False)

# print('Reference: Magenta | Vina Pose: Cyan')
# print ('Pose: {} | Score: {}'.format(results[0].GetProp('Pose'),results[0].GetProp('Score')))

# view.addModel(p,'mol')
# x = view.getModel()
# x.setStyle({},{'stick':{'colorscheme':'cyanCarbon','radius':0.2}})

view.zoomTo()
view.show()
```



```
view = py3Dmol.view()
view.removeAllModels()
view.setViewStyle({'style':'outline','color':'black','width':0.1})

view.addModel(open(hit_Protein,'r').read(),format='pdb')
Prot=view.getModel()
Prot.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'white'}})
view.addSurface(py3Dmol.VDW,{'opacity':0.8,'color':'white'})


view.addModel(open(hit_Ligand,'r').read(),format='mol2')
ref_m = view.getModel()
ref_m.setStyle({},{'stick':{'colorscheme':'magentaCarbon','radius':0.2}})
# ref_m.setStyle({},{'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})
# ref_m.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})


# results=Chem.SDMolSupplier('1AZ8_lig_vina_out.sdf')

# p=Chem.MolToMolBlock(results[0],False)

# print('Reference: Magenta | Vina Pose: Cyan')
# print ('Pose: {} | Score: {}'.format(results[0].GetProp('Pose'),results[0].GetProp('Score')))

# view.addModel(p,'mol')
# x = view.getModel()
# x.setStyle({},{'stick':{'colorscheme':'cyanCarbon','radius':0.2}})

view.zoomTo()
view.show()
```

```
view = py3Dmol.view()
view.removeAllModels()
view.setViewStyle({'style':'outline','color':'black','width':0.1})

view.addModel(open(no_Protein,'r').read(),format='pdb')
Prot=view.getModel()
Prot.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'white'}})
view.addSurface(py3Dmol.VDW,{'opacity':0.8,'color':'white'})


view.addModel(open(no_Ligand,'r').read(),format='mol2')
ref_m = view.getModel()
ref_m.setStyle({},{'stick':{'colorscheme':'magentaCarbon','radius':0.2}})
# ref_m.setStyle({},{'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})
# ref_m.setStyle({'cartoon':{'arrows':True, 'tubes':True, 'style':'oval', 'color':'red'}})


# results=Chem.SDMolSupplier('1AZ8_lig_vina_out.sdf')

# p=Chem.MolToMolBlock(results[0],False)

# print('Reference: Magenta | Vina Pose: Cyan')
# print ('Pose: {} | Score: {}'.format(results[0].GetProp('Pose'),results[0].GetProp('Score')))

# view.addModel(p,'mol')
# x = view.getModel()
# x.setStyle({},{'stick':{'colorscheme':'cyanCarbon','radius':0.2}})

view.zoomTo()
view.show()
```