In [18]:

```
import pandas as pd
import numpy as np
import pickle
from tensorflow.keras.models import load_model
import warnings
from sklearn.metrics import log_loss
from tqdm import tqdm
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix, mean_squared_error
import tensorflow as tf
from tensorflow.keras import layers,regularizers,Sequential,backend,callbacks,optimizer
s,metrics, Model,losses
from sklearn.decomposition import PCA
```

In [19]:

```
# !unzip -q /content/gene_fet_auto_encd.zip
# !unzip -q /content/cell_fet_auto_encd.zip
```

In [20]:

```
p_min = 0.0005
p_max = 0.9995
def logloss(y_true, y_pred):
    y_pred = tf.clip_by_value(y_pred,p_min,p_max)
    return -backend.mean(y_true*backend.log(y_pred) + (1-y_true)*backend.log(1-y_pred))
```

In [21]:

```
def final_fun_1(test_features):
    categorical = 'cp_dose'
    le = pickle.load(open('labelencoder.pkl','rb'))
    test_features[categorical] = le.transform(test_features[categorical])

    scaler = pickle.load(open('transform.pkl','rb'))
    data_test = scaler.transform(test_features.drop('cp_type',axis=1).iloc[:,2:])


    std_scaler = pickle.load(open('standardscaler.pkl', 'rb'))
    data_test =  np.concatenate((std_scaler.transform(test_features.drop('cp_type',axis
=1).iloc[:,:2]),data_test),axis=1)

    c_f = test_features.drop('cp_type',axis=1).columns.str.contains('c-')
    cell_auto_encd = load_model('cell_fet_auto_encd')

    cells_test = data_test[:,c_f]
    ae_cells_test = cell_auto_encd.encoder(cells_test).numpy()

    g_f = test_features.drop('cp_type',axis=1).columns.str.contains('g-')
    gene_auto_encd = load_model('gene_fet_auto_encd')

    genes_test = data_test[:,g_f]
    ae_genes_test = gene_auto_encd.encoder(genes_test).numpy()

    data_test = np.concatenate((data_test[:,~(c_f+g_f)],ae_genes_test,ae_cells_test),ax
is=1)

    dependencies = {'logloss': logloss}
    # getting more features using auto-encoder
    model = load_model('AutoEncoded_seed_18_fold_4.h5',custom_objects=dependencies)

    # Run prediction
    y_pred = model.predict(data_test)/(4*18)

    columns = pickle.load(open('target_columns.pkl','rb'))
    pred_data = pd.DataFrame(y_pred, columns = columns)

    return pred_data
```

In [22]:

```
# !pip3 install scikit-learn==0.22.2.post1
```

In [23]:

```
test = pd.read_csv('test_features - Copy.csv')
test_copy = test.copy()
test = test.drop(['sig_id'],axis=1)
```

In [24]:

```
#calling first function
pred = final_fun_1(test)
# print(pred)
```

In [25]:

```
# pred.insert(loc=0,column='sig_id',value = test_copy['sig_id'])
# pred.head()
pred.insert(loc=0,column='sig_id',value = test_copy['sig_id'])
```

In [26]:

```
pred.head()
```

Out[26]:

	sig_id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylchol
0	id_0004d9e33	0.000018	0.000014	0.000028	0.000078

1 rows × 207 columns

In [ ]:

```
pred.to_csv('final_csv.csv')
```

Deployment Video Link

- https://drive.google.com/file/d/19pznBsglSZzQpk-eba7HAlwPChQDu8ZA/view?usp=sharing (https://drive.google.com/file/d/19pznBsglSZzQpk-eba7HAlwPChQDu8ZA/view?usp=sharing)

Deployment Code

In [ ]:

```
#importing libraries
import pandas as pd
import numpy as np
import pickle
from tensorflow.keras.models import load_model
import warnings
from sklearn.metrics import log_loss
from tqdm import tqdm
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix, mean_squared_error
import tensorflow as tf
from tensorflow.keras import layers,regularizers,Sequential,backend,callbacks,optimizer
s,metrics, Model,losses
from sklearn.decomposition import PCA
from flask import Flask, request, render_template, send_file
import zipfile

app = Flask(__name__)

#loading preprocessed files
columns = pickle.load(open('target_columns.pkl','rb'))

@app.route("/")
def home():
    return render_template('index.html')

@app.route('/predict_moa', methods = ['GET','POST'])
def predict_moa():
    try:

        file = request.files['search_file']
        test_features = pd.read_csv(file, header=0)
        test_copy = test_features.copy()
        test_features = test_features.drop(['sig_id'],axis=1)

        p_min = 0.0005
        p_max = 0.9995
        def logloss(y_true, y_pred):
            y_pred = tf.clip_by_value(y_pred,p_min,p_max)
            return -backend.mean(y_true*backend.log(y_pred) + (1-y_true)*backend.log(1-
y_pred))

        categorical = 'cp_dose'
        le = pickle.load(open('labelencoder.pkl','rb'))
        test_features[categorical] = le.transform(test_features[categorical])

        scaler = pickle.load(open('transform.pkl','rb'))
        data_test = scaler.transform(test_features.drop('cp_type',axis=1).iloc[:,2:])

        std_scaler = pickle.load(open('standardscaler.pkl', 'rb'))
        data_test =  np.concatenate((std_scaler.transform(test_features.drop('cp_type',
axis=1).iloc[:,:2]),data_test),axis=1)
```

```
        c_f = test_features.drop('cp_type',axis=1).columns.str.contains('c-')
        cell_auto_encd = load_model('cell_fet_auto_encd')

        cells_test = data_test[:,c_f]
        ae_cells_test = cell_auto_encd.encoder(cells_test).numpy()

        g_f = test_features.drop('cp_type',axis=1).columns.str.contains('g-')
        gene_auto_encd = load_model('gene_fet_auto_encd')

        genes_test = data_test[:,g_f]
        ae_genes_test = gene_auto_encd.encoder(genes_test).numpy()

        data_test = np.concatenate((data_test[:,~(c_f+g_f)],ae_genes_test,ae_cells_test
),axis=1)

        dependencies = {'logloss': logloss}
        # getting more features using auto-encoder
        model = load_model('AutoEncoded_seed_18_fold_4.h5',custom_objects=dependencies)

        # Run prediction
        y_pred = model.predict(data_test)/(4*18)

        columns = pickle.load(open('target_columns.pkl','rb'))
        pred_data = pd.DataFrame(y_pred, columns = columns)


        pred_data.insert(loc=0,column='sig_id',value = test_copy['sig_id'])
        pred_data.to_csv('final_csv.csv')
        #zipping prediction and probability prediction
        # zipf = zipfile.ZipFile('Predictions.zip','w', zipfile.ZIP_DEFLATED)
        # zipf.write('pred_data.csv')
        # zipf.write('pred_prob_data.csv')
        # zipf.close()

        return send_file('final_csv.csv', mimetype='csv', as_attachment=True, attachmen
t_filename='moa_final.csv')

    #if file is not same as in instruction then exception will be thrown
    except Exception as e:
        # print('The Exception message is: ',e)
        return render_template('index.html',error=e)

@app.route('/Important Instructions')
def imp_instruction():
    return render_template('instruction.html')

@app.route('/download')
def download_file():
    return send_file('sample_input.csv', as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```