PariSa Jangani

In [3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

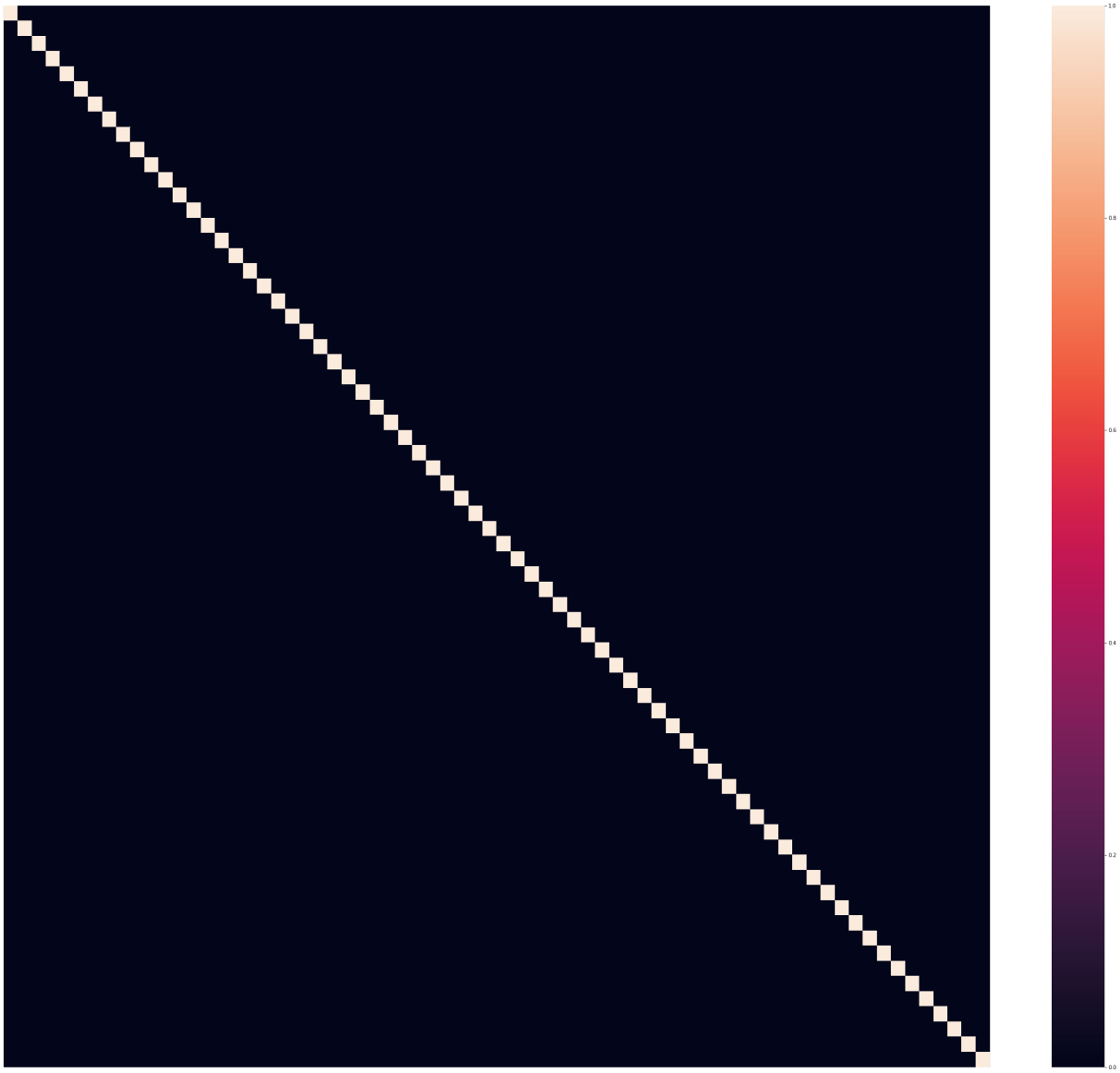
```
Mounted at /content/drive
```

In [4]:
```
from numpy.core.numeric import NaN
import pandas as pd
import numpy as np
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
dfeature= pd.read_csv('/content/drive/My Drive/tox21_global_cdf_rdkit.csv')
dlabel= pd.read_csv('/content/drive/My Drive/tox21label.csv')
data0=pd.concat([dfeature,dlabel],axis=1)

data1 = data0.iloc[:,1:]

dataf= data1.dropna(how='any')#data was cleaned
#dataf=datafff.drop(index=3361)

data1 = data0.iloc[:,1:]
dataf= data1.dropna(how='any')#data was cleaned
#dataf=datafff.drop(index=3361)

xdata = dataf.iloc[:,:-2] #features
ydata = dataf.iloc[:,-2:] # smile & target label
ydata=pd.DataFrame(ydata)
#dlabel.value_counts(ydata['SR-ARE']==1)
```

In [6]:
```
from pandas._libs.hashtable import value_count
from sklearn.model_selection import train_test_split
X_trainingdata, X_test, y_trainingdata, y_test =train_test_split(xdata,ydata, stra
```

In [8]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=70)
principalComponents = pca.fit_transform(X_trainingdata)
X_trainingdata_PCA = pd.DataFrame(data = principalComponents)
X_trainingdata_PCA
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(43,38))
a=sns.heatmap(X_trainingdata_PCA.corr(),xticklabels=False, yticklabels=False)
```



In [9]:
```
print(np.sum(pca.explained_variance_ratio_))
```

```
0.9509928062647196
```

In [ ]:
```
y_trainingdata_PCA=pd.DataFrame(data =y_trainingdata.to_numpy()[:,1],columns=['SR-
y_trainingdata_PCA
```

In [11]:
```
from sklearn.ensemble import AdaBoostClassifier
clf=AdaBoostClassifier()#clf==grid?
clf.fit(X_trainingdata,y_trainingdata['SR-ARE'])
cptrain=clf.score(X_trainingdata,y_trainingdata['SR-ARE'])
print('training score = ',cptrain)
```

```
training score =  0.8553772766695577
```

In [12]:
```
cptest=clf.score(X_test,y_test['SR-ARE'])
print('test score = ',cptest)
```

```
test score =  0.8551604509973981
```

In [47]:
```
from sklearn.ensemble import AdaBoostClassifier
clf=AdaBoostClassifier()#clf==grid?
#clf.fit(X_trainingdata_PCA,y_trainingdata_PCA['SR-ARE'])
clf.fit(X_trainingdata_PCA.to_numpy().astype('float'),(np.transpose(y_trainingdata
cptrain=clf.score(X_trainingdata_PCA.to_numpy().astype('float'),(np.transpose(y_tr
#cptrain=clf.score(X_trainingdata_PCA,y_trainingdata_PCA['SR-ARE'])
print('training score = ',cptrain)
```

```
training score =  0.8512575888985255
```

In [48]:
```
cptest=clf.score(X_test_pca,y_test['SR-ARE'])
print('test score = ',cptest)
```

```
test score =  0.8464874241110147
```

In [49]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix,plot_confusion_matrix
Adabost = AdaBoostClassifier()
param_grid={'n_estimators':np.arange(1,100,5)}
grid=GridSearchCV(Adabost,param_grid=param_grid,cv=10,scoring='balanced_accuracy',
grid.fit(X_trainingdata_PCA.to_numpy().astype('float'),(np.transpose(y_trainingdat
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

```
best mean cv score=  0.5843448914125893
best parameters=  {'n_estimators': 96}
```

In [18]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix,plot_confusion_matrix
Adabost = AdaBoostClassifier()
param_grid={'n_estimators':np.arange(1,100,5)}
grid=GridSearchCV(Adabost,param_grid=param_grid,cv=10,scoring='balanced_accuracy',
grid.fit(X_trainingdata.to_numpy().astype('float'),(np.transpose((y_trainingdata['
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

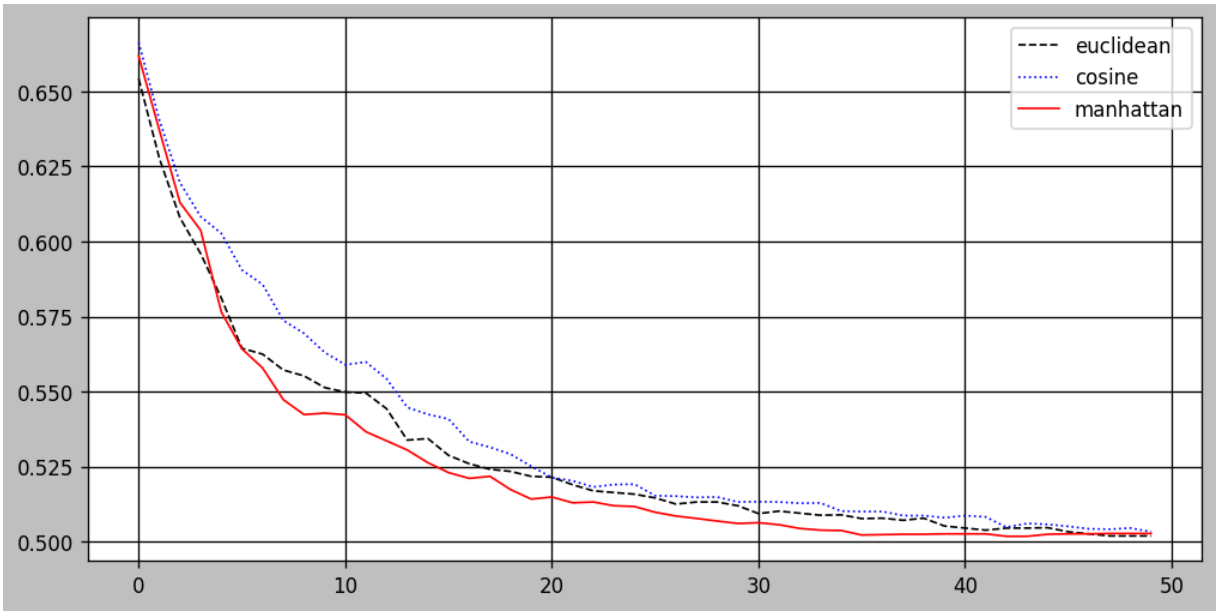
```
best mean cv score=  0.5917590939074804
best parameters=  {'n_estimators': 91}
```

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix,plot_confusion_matrix
knn = KNeighborsClassifier()
param_grid={'n_neighbors':np.arange(1,100,2),'metric':['euclidean','cosine','manha
grid=GridSearchCV(knn,param_grid=param_grid,cv=10,scoring='balanced_accuracy',retu
grid.fit(X_trainingdata_PCA.to_numpy().astype('float'),np.transpose(y_trainingdata
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

```
best mean cv score=  0.6663779188958904
best parameters=  {'metric': 'cosine', 'n_neighbors': 1}
```

In [21]:
```
knn_cv_results=grid.cv_results_
```

In [50]:
```
plt.style.use('grayscale')
plt.figure(figsize=(10,5),dpi=120)
plt.plot(np.arange(0,50,1),knn_cv_results['mean_test_score'][0:50],lw=1,label='euc
plt.plot(np.arange(0,50,1),knn_cv_results['mean_test_score'][50:100],lw=1,label='c
plt.plot(np.arange(0,50,1),knn_cv_results['mean_test_score'][100:150],lw=1,label='
plt.legend()
plt.grid()
plt.show()
import matplotlib.pyplot as plt

fig = plt.figure()
fig.canvas.manager.full_screen_toggle() # toggle fullscreen mode
fig.show()
```




```
<Figure size 432x288 with 0 Axes>
```

In [22]:
```
X_test_pca=pd.DataFrame(data=pca.transform(X_test))
grid.score(X_test_pca,y_test['SR-ARE'])
```

Out[22]: 0.6641646201378436

In [23]:
```
y_pred_knn=grid.best_estimator_.predict(X_test_pca)
print(confusion_matrix(y_test['SR-ARE'],np.transpose(np.matrix(y_pred_knn))))
```

```
[[879  92]
 [105  77]]
```

In [39]:
```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
DecisionTree = DecisionTreeClassifier()
param_grid={'criterion':['gini','entropy'],'max_depth':[5,10,15,20,25,30]}
grid=GridSearchCV(DecisionTree,param_grid=param_grid,cv=8,scoring='balanced_accura
grid.fit(X_trainingdata_PCA.to_numpy().astype('float'),np.transpose(y_trainingdata
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

```
best mean cv score=  0.6023835107658693
best parameters=  {'criterion': 'entropy', 'max_depth': 25}
```

In [40]:
```
y_pred_dt=grid.best_estimator_.predict(X_test_pca)
print(confusion_matrix(y_test['SR-ARE'],np.transpose(np.matrix(y_pred_dt))))
print(grid.score(X_test_pca,y_test['SR-ARE']))
```

```
[[881  90]
 [108  74]]
0.6569527280134901
```

In [42]:
```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
DecisionTree = DecisionTreeClassifier()
param_grid={'criterion':['gini','entropy'],'max_depth':[12,17,18,19,21,25,30]}
grid=GridSearchCV(DecisionTree,param_grid=param_grid,cv=10,scoring='balanced_accur
grid.fit(X_trainingdata.to_numpy().astype('float'),y_trainingdata['SR-ARE'])
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

```
best mean cv score=  0.6475554354696246
best parameters=  {'criterion': 'entropy', 'max_depth': 19}
```

In [43]:
```
y_pred_dt2=grid.best_estimator_.predict(X_test)
print(confusion_matrix(y_test['SR-ARE'],np.transpose(np.matrix(y_pred_dt2))))
print(grid.score(X_test,y_test['SR-ARE']))
```

```
[[866 105]
 [112  70]]
0.6382397211439436

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but DecisionTreeClassifier was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but DecisionTreeClassifier was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

In [53]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
RFC = RandomForestClassifier()
param_grid={'n_estimators':[300],'criterion':['gini','entropy'],'max_depth':[3,4,5
grid=GridSearchCV(RFC,param_grid=param_grid,cv=5,scoring='balanced_accuracy',retur
grid.fit(X_trainingdata.to_numpy().astype('float'),y_trainingdata['SR-ARE'])
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

```
best mean cv score=  0.5170351891524748
best parameters=  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'aut
o', 'n_estimators': 300}
```

In [54]:
```
y_pred_RFC=grid.best_estimator_.predict(X_test)
print(confusion_matrix(y_test['SR-ARE'],np.transpose(np.matrix(y_pred_RFC))))
print(grid.score(X_test,y_test['SR-ARE']))
```

```
[[969   2]
 [172  10]]
0.5264426613551227

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but RandomForestClassifier was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but RandomForestClassifier was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

In [55]:
```
from sklearn.svm import SVC
param_grid={'kernel':['poly'],'degree':[3,4,5,6,7,8]}
grid=GridSearchCV(SVC(),param_grid=param_grid,cv=5,scoring='balanced_accuracy',ret
grid.fit(X_trainingdata.to_numpy().astype('float'),(np.transpose((y_trainingdata['
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

```
best mean cv score=  0.6718793853873646
best parameters=  {'degree': 5, 'kernel': 'poly'}
```

In [56]:
```
y_pred_svm=grid.best_estimator_.predict(X_test)
print(confusion_matrix(y_test['SR-ARE'],np.transpose(np.matrix(y_pred_svm))))
print(grid.score(X_test,y_test['SR-ARE']))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but SVC was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"

[[916  55]
 [113  69]]
0.6612391213318093

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has
feature names, but SVC was fitted without feature names
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

In [57]:
```
from xgboost import XGBClassifier
import warnings
warnings.simplefilter('ignore')
param_grid={'n_estimators':[200,700],'max_depth':[3,4,5,6,7],'eval_metric':['mlogl
grid=GridSearchCV(XGBClassifier(),param_grid=param_grid,cv=5,scoring='balanced_acc
grid.fit(X_trainingdata.to_numpy().astype('float'),(np.transpose((y_trainingdata['
print("best mean cv score= ",grid.best_score_)
print("best parameters= ",grid.best_params_)
```

```
best mean cv score=  0.6545755294907957
best parameters=  {'eval_metric': 'mlogloss', 'max_depth': 5, 'n_estimators':
700}
```

In [ ]:
```
y_pred_xgbc=grid.best_estimator_.predict(X_test)
print(confusion_matrix(y_test['SR-ARE'] ,np.transpose(np.matrix(y_pred_xgbc))))
print(grid.score(X_test,y_test['SR-ARE']))
```