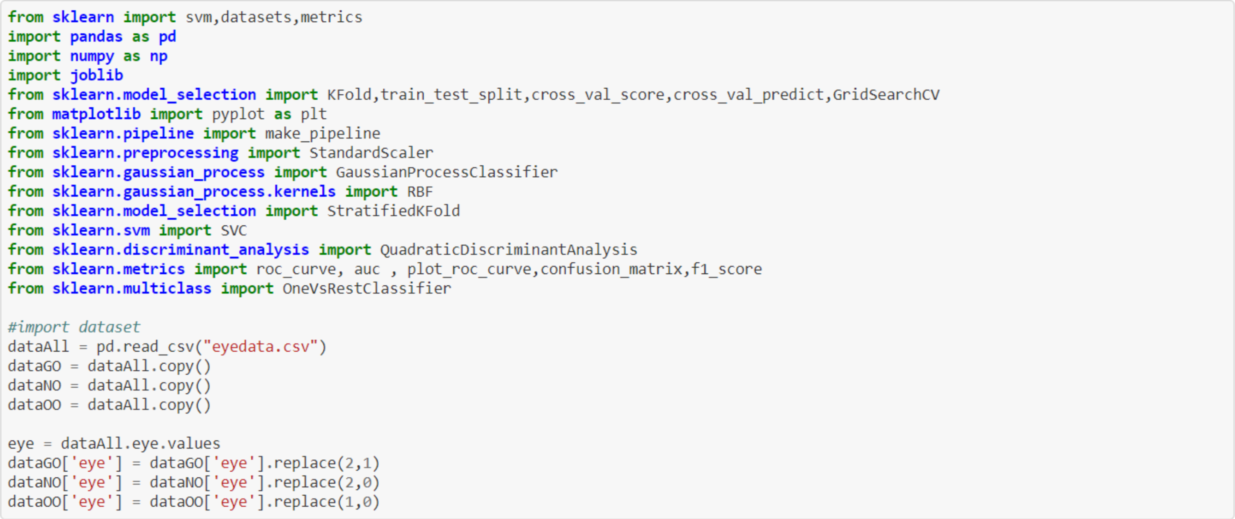
Machine Learning Code Explanation

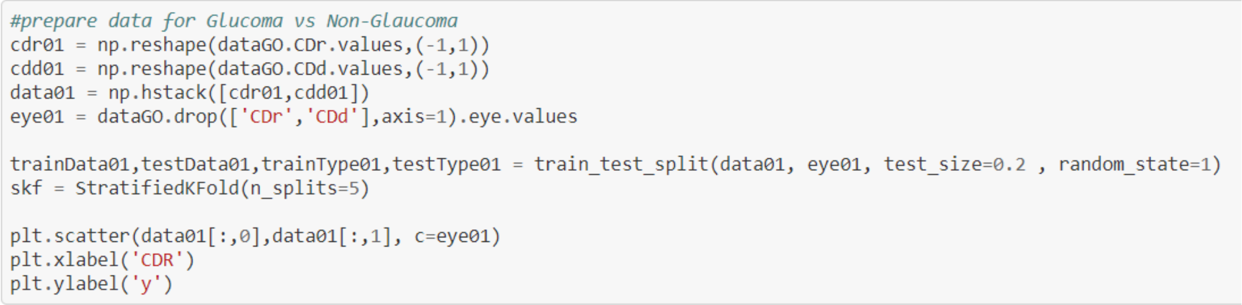
machine model QDA

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L1-L26



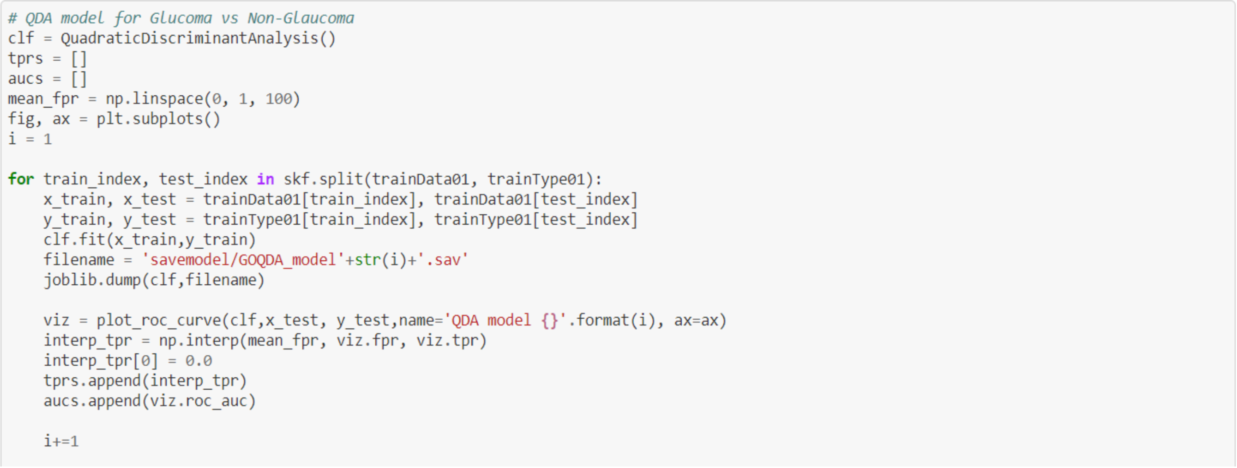
```
from sklearn import svm,datasets,metrics
import pandas as pd
import numpy as np
import joblib
from sklearn.model_selection import KFold,train_test_split,cross_val_score,cross_val_predict,GridSearchCV
from matplotlib import pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import roc_curve, auc , plot_roc_curve,confusion_matrix,f1_score
from sklearn.multiclass import OneVsRestClassifier

#import dataset
dataAll = pd.read_csv("eyedata.csv")
dataGO = dataAll.copy()
dataNO = dataAll.copy()
dataOO = dataAll.copy()

eye = dataAll.eye.values
dataGO['eye'] = dataGO['eye'].replace(2,1)
dataNO['eye'] = dataNO['eye'].replace(2,0)
dataOO['eye'] = dataOO['eye'].replace(1,0)
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L29-L40

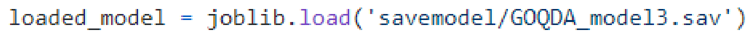


```
#prepare data for Glucoma vs Non-Glaucoma
cdr01 = np.reshape(dataGO.CDr.values,(-1,1))
cdd01 = np.reshape(dataGO.CDd.values,(-1,1))
data01 = np.hstack([cdr01,cdd01])
eye01 = dataGO.drop(['CDr','CDd'],axis=1).eye.values

trainData01,testData01,trainType01,testType01 = train_test_split(data01, eye01, test_size=0.2 , random_state=1)
skf = StratifiedKFold(n_splits=5)

plt.scatter(data01[:,0],data01[:,1], c=eye01)
plt.xlabel('CDR')
plt.ylabel('y')
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L43-L64

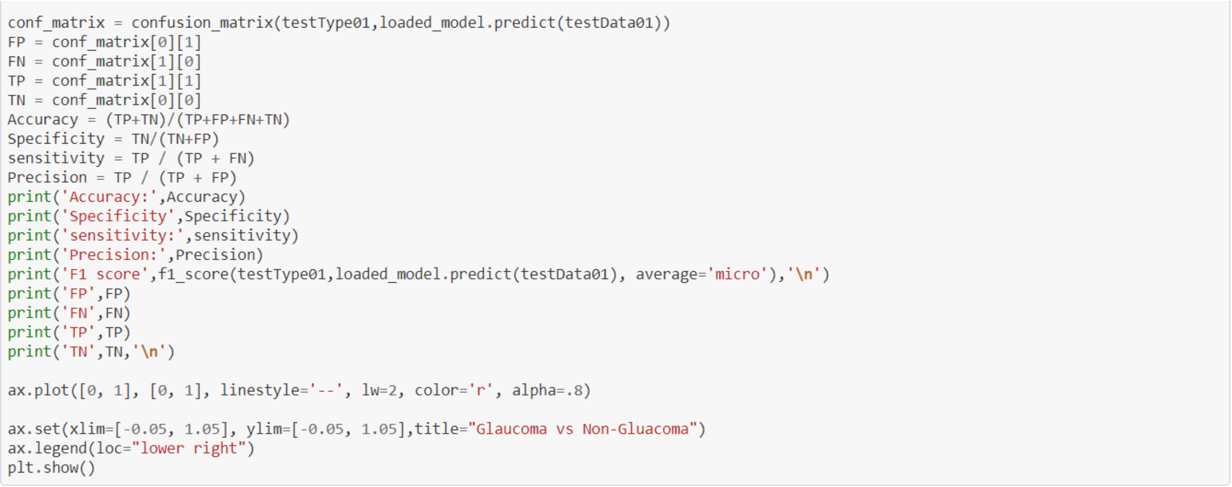


```
# QDA model for Glucoma vs Non-Glaucoma
clf = QuadraticDiscriminantAnalysis()
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
fig, ax = plt.subplots()
i = 1

for train_index, test_index in skf.split(trainData01, trainType01):
    x_train, x_test = trainData01[train_index], trainData01[test_index]
    y_train, y_test = trainType01[train_index], trainType01[test_index]
    clf.fit(x_train,y_train)
    filename = 'savemodel/GOQDA_model'+str(i)+'.sav'
    joblib.dump(clf,filename)

    viz = plot_roc_curve(clf,x_test, y_test,name='QDA model {}'.format(i), ax=ax)
    interp_tpr = np.interp(mean_fpr, viz.fpr, viz.tpr)
    interp_tpr[0] = 0.0
    tprs.append(interp_tpr)
    aucs.append(viz.roc_auc)

    i+=1
```



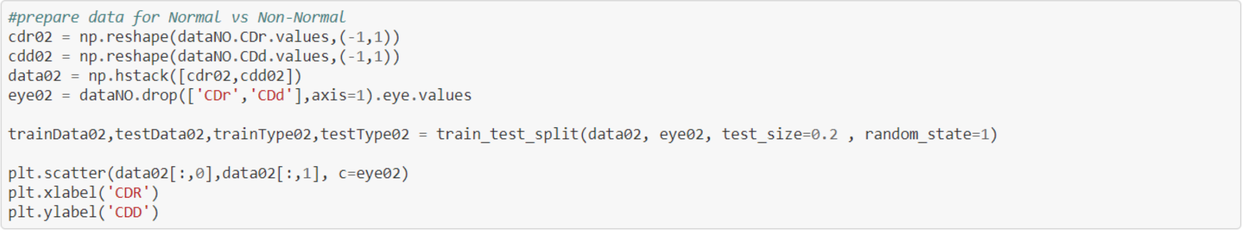
```
loaded_model = joblib.load('savemodel/GOQDA_model3.sav')
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L68-L91



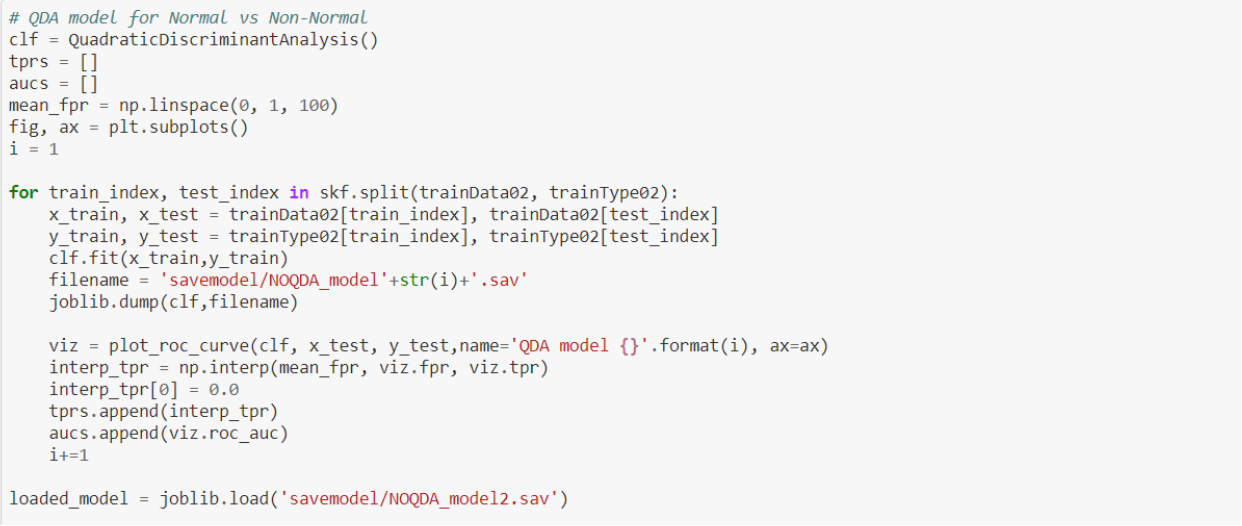
```
conf_matrix = confusion_matrix(testType01,loaded_model.predict(testData01))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType01,loaded_model.predict(testData01), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')

ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', alpha=.8)

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],title="Glaucoma vs Non-Gluacoma")
ax.legend(loc="lower right")
plt.show()
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L94-L104



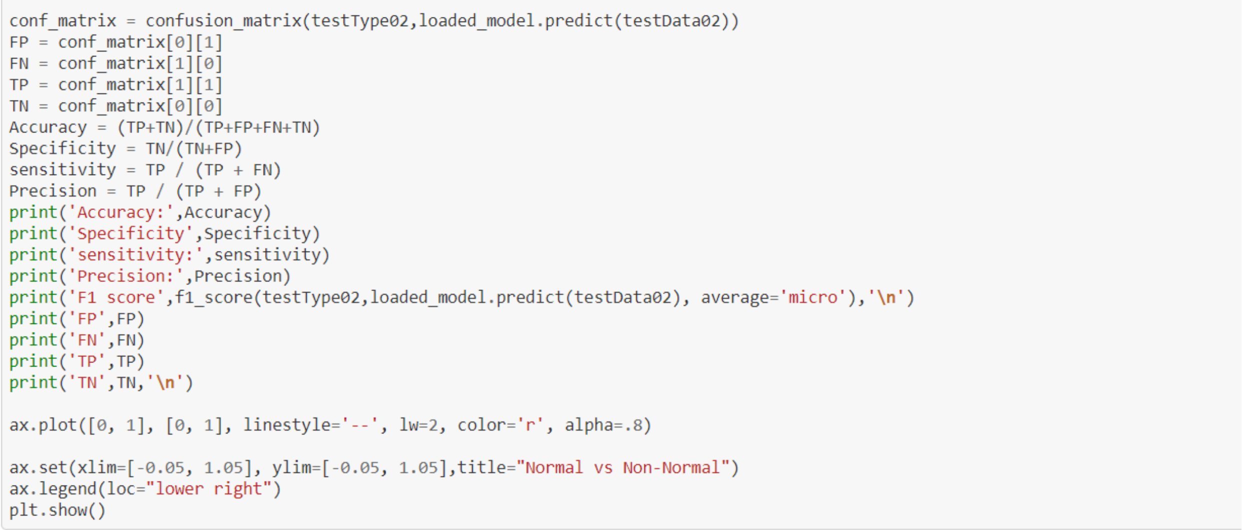
```
#prepare data for Normal vs Non-Normal
cdr02 = np.reshape(dataNO.CDr.values,(-1,1))
cdd02 = np.reshape(dataNO.CDd.values,(-1,1))
data02 = np.hstack([cdr02,cdd02])
eye02 = dataNO.drop(['CDr','CDd'],axis=1).eye.values

trainData02,testData02,trainType02,testType02 = train_test_split(data02, eye02, test_size=0.2 , random_state=1)

plt.scatter(data02[:,0],data02[:,1], c=eye02)
plt.xlabel('CDR')
plt.ylabel('CDD')
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L107-L129



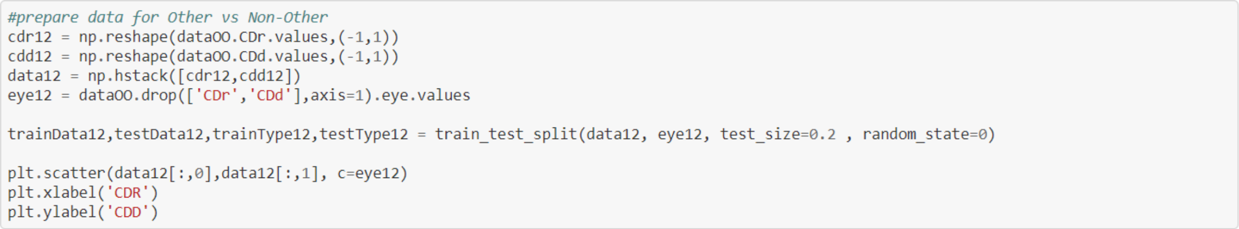
```python
# QDA model for Normal vs Non-Normal
clf = QuadraticDiscriminantAnalysis()
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
fig, ax = plt.subplots()
i = 1

for train_index, test_index in skf.split(trainData02, trainType02):
    x_train, x_test = trainData02[train_index], trainData02[test_index]
    y_train, y_test = trainType02[train_index], trainType02[test_index]
    clf.fit(x_train,y_train)
    filename = 'savemodel/NOQDA_model'+str(i)+'.sav'
    joblib.dump(clf,filename)

    viz = plot_roc_curve(clf, x_test, y_test,name='QDA model {}'.format(i), ax=ax)
    interp_tpr = np.interp(mean_fpr, viz.fpr, viz.tpr)
    interp_tpr[0] = 0.0
    tprs.append(interp_tpr)
    aucs.append(viz.roc_auc)
    i+=1

loaded_model = joblib.load('savemodel/NOQDA_model2.sav')
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L131-L154



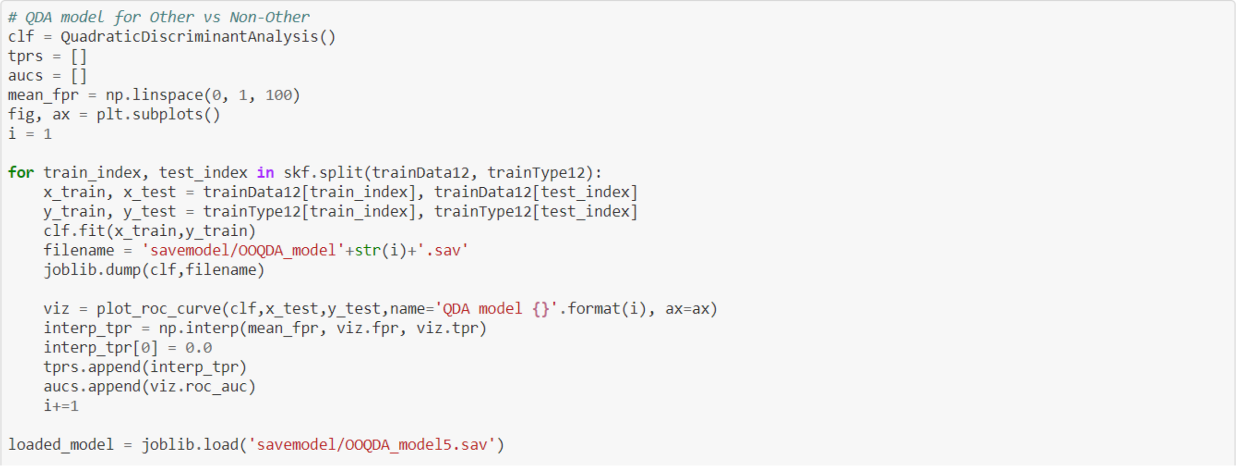
```python
conf_matrix = confusion_matrix(testType02,loaded_model.predict(testData02))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType02,loaded_model.predict(testData02), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')

ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', alpha=.8)

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],title="Normal vs Non-Normal")
ax.legend(loc="lower right")
plt.show()
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L157-L167



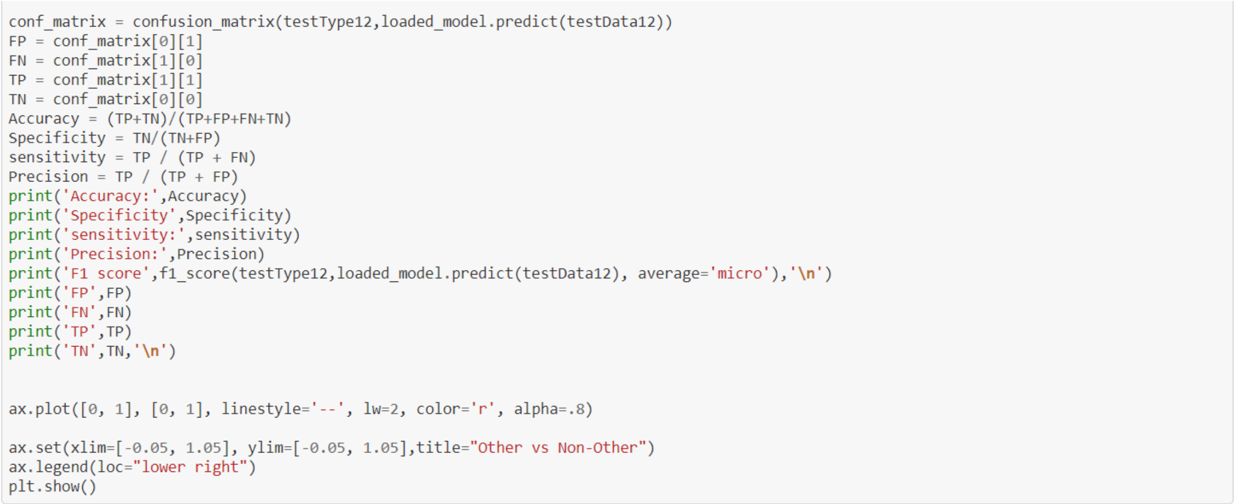
```
#prepare data for Other vs Non-Other
cdr12 = np.reshape(dataOO.CDr.values,(-1,1))
cdd12 = np.reshape(dataOO.CDd.values,(-1,1))
data12 = np.hstack([cdr12,cdd12])
eye12 = dataOO.drop(['CDr','CDd'],axis=1).eye.values

trainData12,testData12,trainType12,testType12 = train_test_split(data12, eye12, test_size=0.2 , random_state=0)

plt.scatter(data12[:,0],data12[:,1], c=eye12)
plt.xlabel('CDR')
plt.ylabel('CDD')
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L170-L192



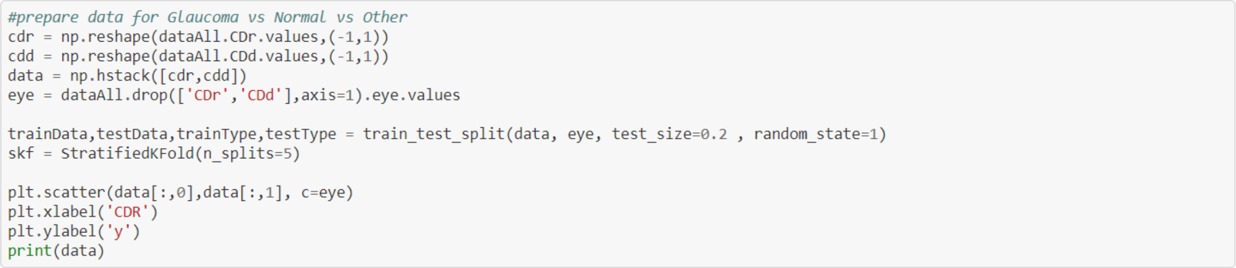
```
# QDA model for Other vs Non-Other
clf = QuadraticDiscriminantAnalysis()
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
fig, ax = plt.subplots()
i = 1

for train_index, test_index in skf.split(trainData12, trainType12):
    x_train, x_test = trainData12[train_index], trainData12[test_index]
    y_train, y_test = trainType12[train_index], trainType12[test_index]
    clf.fit(x_train,y_train)
    filename = 'savemodel/OOQDA_model'+str(i)+'.sav'
    joblib.dump(clf,filename)

    viz = plot_roc_curve(clf,x_test,y_test,name='QDA model {}'.format(i), ax=ax)
    interp_tpr = np.interp(mean_fpr, viz.fpr, viz.tpr)
    interp_tpr[0] = 0.0
    tprs.append(interp_tpr)
    aucs.append(viz.roc_auc)
    i+=1

loaded_model = joblib.load('savemodel/OOQDA_model5.sav')
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L194-L218



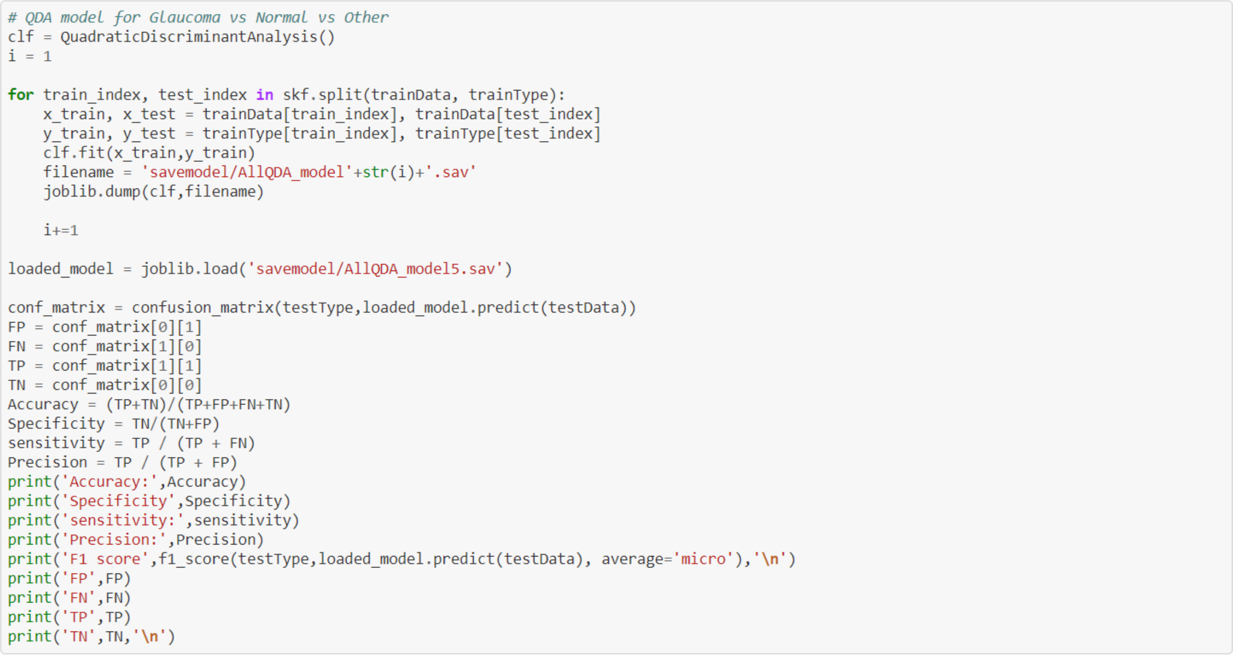
```
conf_matrix = confusion_matrix(testType12,loaded_model.predict(testData12))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType12,loaded_model.predict(testData12), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')


ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', alpha=.8)

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],title="Other vs Non-Other")
ax.legend(loc="lower right")
plt.show()
```

https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L221-L233



```
#prepare data for Glaucoma vs Normal vs Other
cdr = np.reshape(dataAll.CDr.values,(-1,1))
cdd = np.reshape(dataAll.CDd.values,(-1,1))
data = np.hstack([cdr,cdd])
eye = dataAll.drop(['CDr','CDd'],axis=1).eye.values

trainData,testData,trainType,testType = train_test_split(data, eye, test_size=0.2 , random_state=1)
skf = StratifiedKFold(n_splits=5)

plt.scatter(data[:,0],data[:,1], c=eye)
plt.xlabel('CDR')
plt.ylabel('y')
print(data)
```

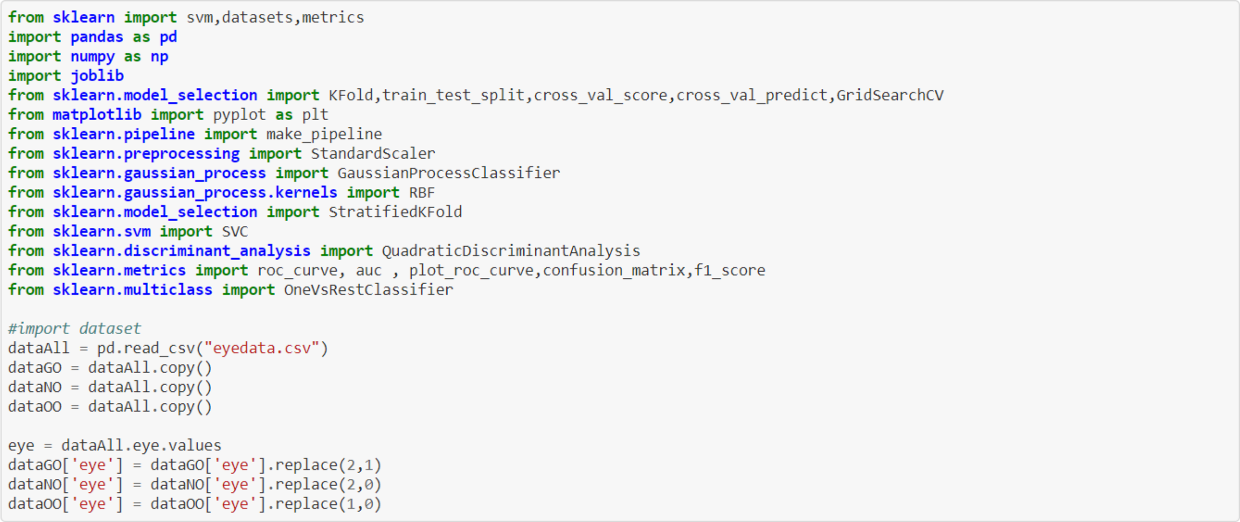
https://github.com/gringgyy/cn240/blob/0f8a772fa9a81421849b95cecb7bffbeab8e4810/machineLearning/MachinemodelQDA.py#L236-L268



```
# QDA model for Glaucoma vs Normal vs Other
clf = QuadraticDiscriminantAnalysis()
i = 1

for train_index, test_index in skf.split(trainData, trainType):
    x_train, x_test = trainData[train_index], trainData[test_index]
    y_train, y_test = trainType[train_index], trainType[test_index]
    clf.fit(x_train,y_train)
    filename = 'savemodel/AllQDA_model'+str(i)+'.sav'
    joblib.dump(clf,filename)

    i+=1

loaded_model = joblib.load('savemodel/AllQDA_model5.sav')

conf_matrix = confusion_matrix(testType,loaded_model.predict(testData))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType,loaded_model.predict(testData), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')
```

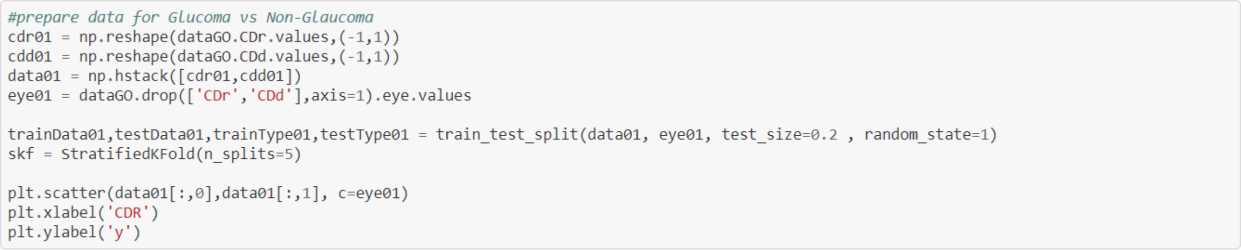
machine model RBF

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L1-L27



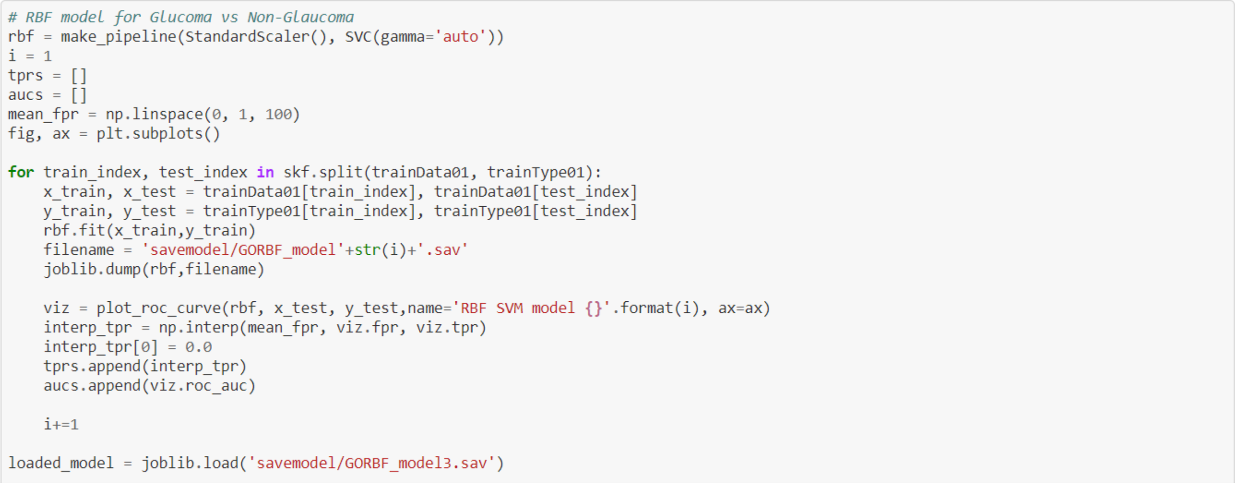
```
from sklearn import svm,datasets,metrics
import pandas as pd
import numpy as np
import joblib
from sklearn.model_selection import KFold,train_test_split,cross_val_score,cross_val_predict,GridSearchCV
from matplotlib import pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import roc_curve, auc , plot_roc_curve,confusion_matrix,f1_score
from sklearn.multiclass import OneVsRestClassifier

#import dataset
dataAll = pd.read_csv("eyedata.csv")
dataGO = dataAll.copy()
dataNO = dataAll.copy()
dataOO = dataAll.copy()

eye = dataAll.eye.values
dataGO['eye'] = dataGO['eye'].replace(2,1)
dataNO['eye'] = dataNO['eye'].replace(2,0)
dataOO['eye'] = dataOO['eye'].replace(1,0)
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L30-L41



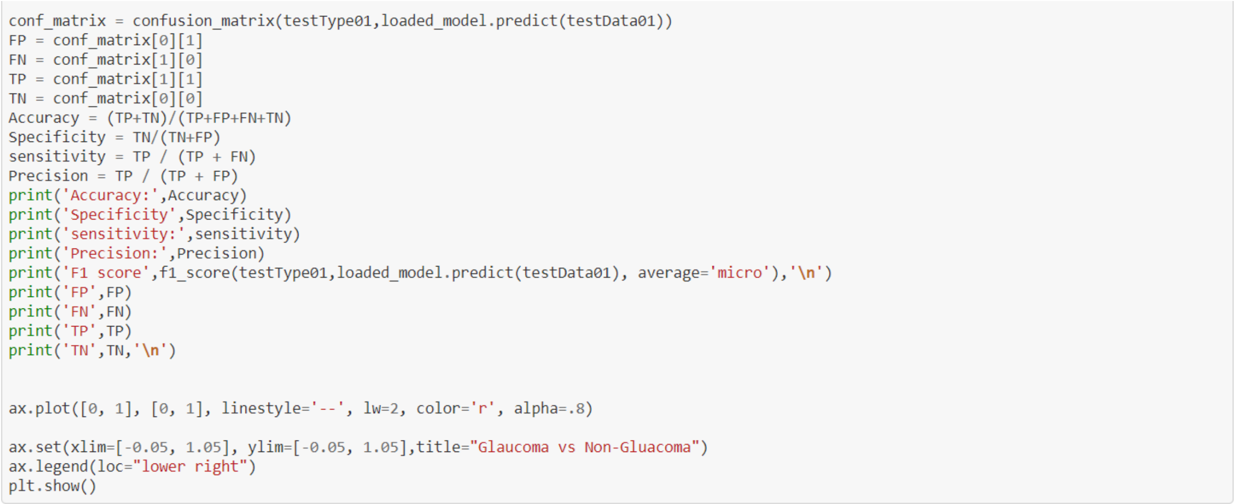
```
#prepare data for Glucoma vs Non-Glaucoma
cdr01 = np.reshape(dataGO.CDr.values,(-1,1))
cdd01 = np.reshape(dataGO.CDd.values,(-1,1))
data01 = np.hstack([cdr01,cdd01])
eye01 = dataGO.drop(['CDr','CDd'],axis=1).eye.values

trainData01,testData01,trainType01,testType01 = train_test_split(data01, eye01, test_size=0.2 , random_state=1)
skf = StratifiedKFold(n_splits=5)

plt.scatter(data01[:,0],data01[:,1], c=eye01)
plt.xlabel('CDR')
plt.ylabel('y')
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L44-L67



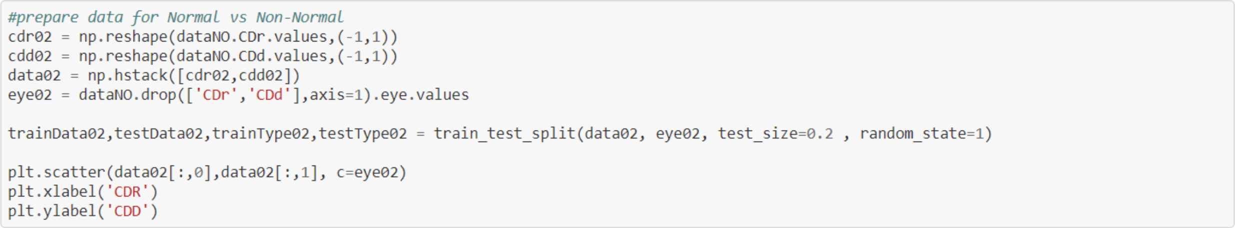
```python
# RBF model for Glucoma vs Non-Glaucoma
rbf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
i = 1
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
fig, ax = plt.subplots()

for train_index, test_index in skf.split(trainData01, trainType01):
    x_train, x_test = trainData01[train_index], trainData01[test_index]
    y_train, y_test = trainType01[train_index], trainType01[test_index]
    rbf.fit(x_train,y_train)
    filename = 'savemodel/GORBF_model'+str(i)+'.sav'
    joblib.dump(rbf,filename)

    viz = plot_roc_curve(rbf, x_test, y_test,name='RBF SVM model {}'.format(i), ax=ax)
    interp_tpr = np.interp(mean_fpr, viz.fpr, viz.tpr)
    interp_tpr[0] = 0.0
    tprs.append(interp_tpr)
    aucs.append(viz.roc_auc)

    i+=1

loaded_model = joblib.load('savemodel/GORBF_model3.sav')
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L69-L92



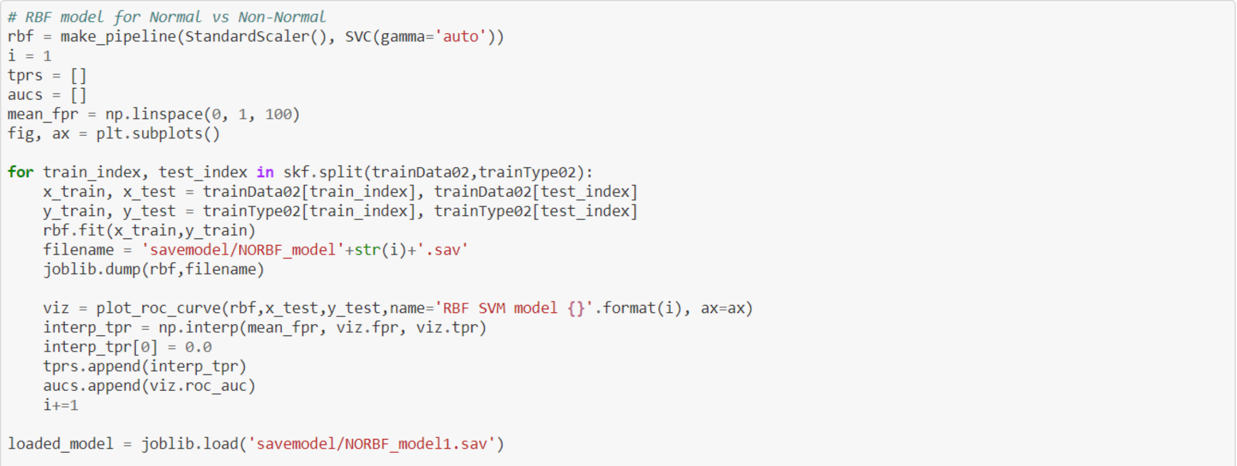
```python
conf_matrix = confusion_matrix(testType01,loaded_model.predict(testData01))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType01,loaded_model.predict(testData01), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')


ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', alpha=.8)

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],title="Glaucoma vs Non-Gluacoma")
ax.legend(loc="lower right")
plt.show()
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L95-L105



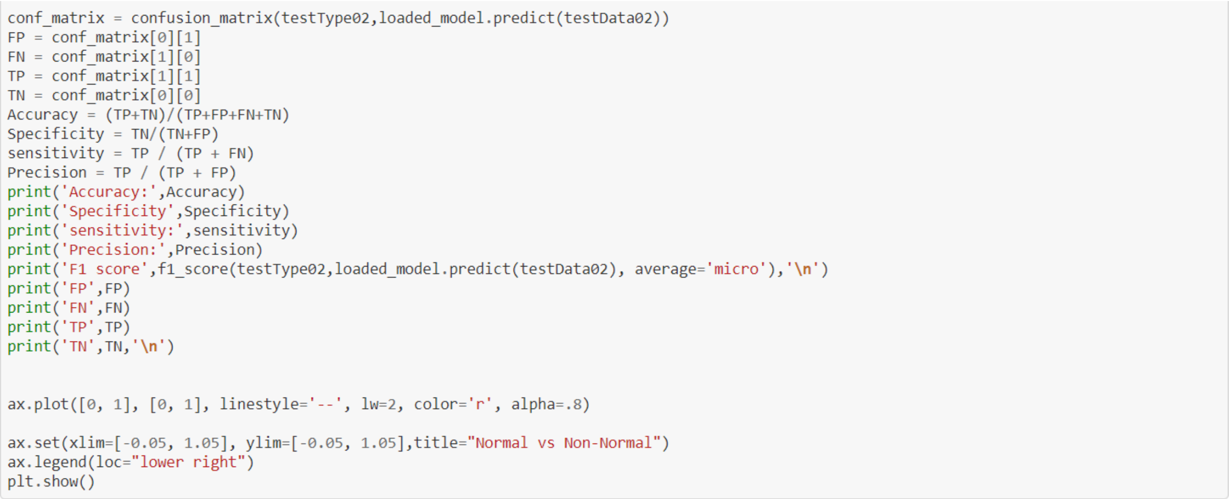
```
#prepare data for Normal vs Non-Normal
cdr02 = np.reshape(dataNO.CDr.values,(-1,1))
cdd02 = np.reshape(dataNO.CDd.values,(-1,1))
data02 = np.hstack([cdr02,cdd02])
eye02 = dataNO.drop(['CDr','CDd'],axis=1).eye.values

trainData02,testData02,trainType02,testType02 = train_test_split(data02, eye02, test_size=0.2 , random_state=1)

plt.scatter(data02[:,0],data02[:,1], c=eye02)
plt.xlabel('CDR')
plt.ylabel('CDD')
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L108-L130



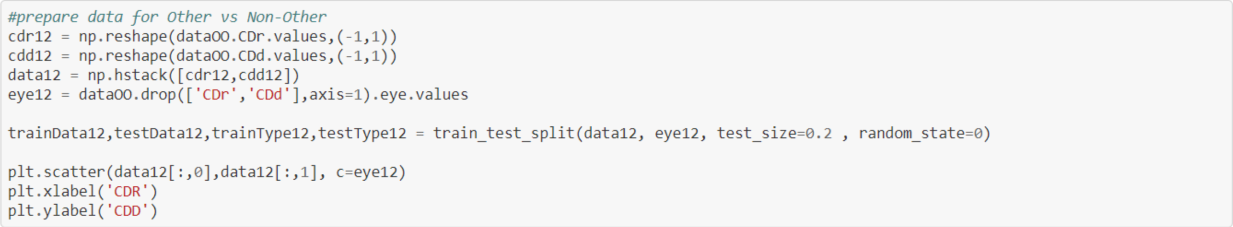
```
# RBF model for Normal vs Non-Normal
rbf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
i = 1
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
fig, ax = plt.subplots()

for train_index, test_index in skf.split(trainData02,trainType02):
    x_train, x_test = trainData02[train_index], trainData02[test_index]
    y_train, y_test = trainType02[train_index], trainType02[test_index]
    rbf.fit(x_train,y_train)
    filename = 'savemodel/NORBF_model'+str(i)+'.sav'
    joblib.dump(rbf,filename)

    viz = plot_roc_curve(rbf,x_test,y_test,name='RBF SVM model {}'.format(i), ax=ax)
    interp_tpr = np.interp(mean_fpr, viz.fpr, viz.tpr)
    interp_tpr[0] = 0.0
    tprs.append(interp_tpr)
    aucs.append(viz.roc_auc)
    i+=1

loaded_model = joblib.load('savemodel/NORBF_model1.sav')
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L132-L155



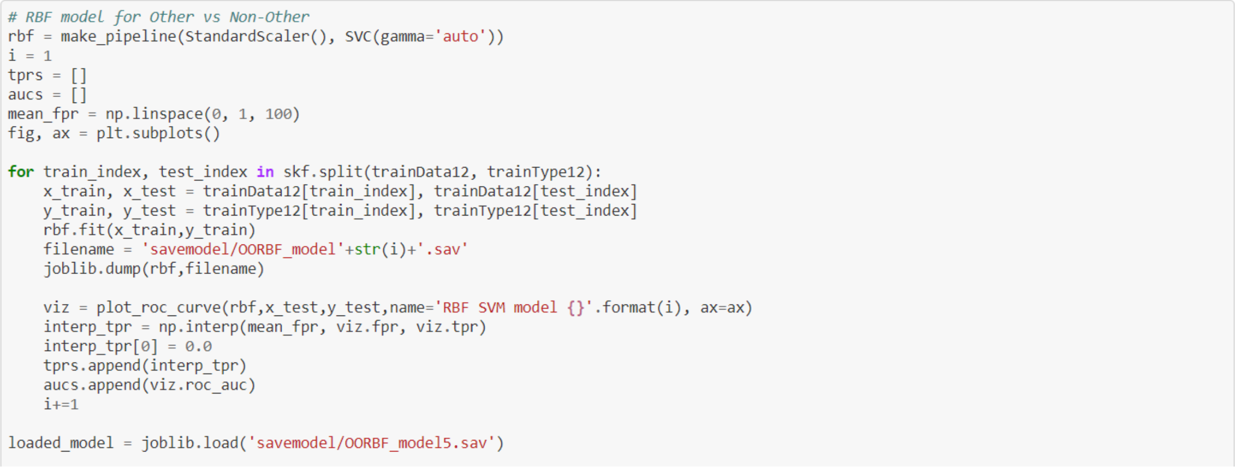
```
conf_matrix = confusion_matrix(testType02,loaded_model.predict(testData02))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType02,loaded_model.predict(testData02), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')


ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', alpha=.8)

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],title="Normal vs Non-Normal")
ax.legend(loc="lower right")
plt.show()
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L158-L168



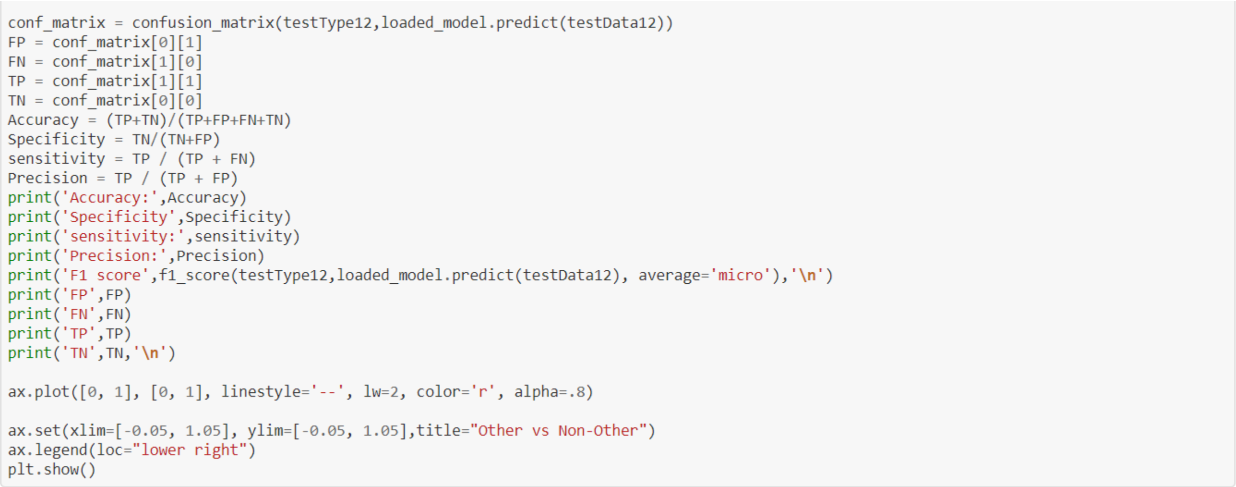
```
#prepare data for Other vs Non-Other
cdr12 = np.reshape(dataOO.CDr.values,(-1,1))
cdd12 = np.reshape(dataOO.CDd.values,(-1,1))
data12 = np.hstack([cdr12,cdd12])
eye12 = dataOO.drop(['CDr','CDd'],axis=1).eye.values

trainData12,testData12,trainType12,testType12 = train_test_split(data12, eye12, test_size=0.2 , random_state=0)

plt.scatter(data12[:,0],data12[:,1], c=eye12)
plt.xlabel('CDR')
plt.ylabel('CDD')
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L171-L193



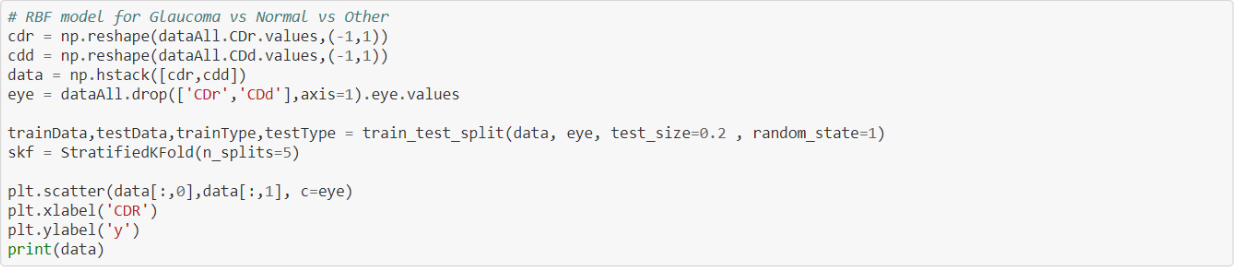
```
# RBF model for Other vs Non-Other
rbf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
i = 1
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)
fig, ax = plt.subplots()

for train_index, test_index in skf.split(trainData12, trainType12):
    x_train, x_test = trainData12[train_index], trainData12[test_index]
    y_train, y_test = trainType12[train_index], trainType12[test_index]
    rbf.fit(x_train,y_train)
    filename = 'savemodel/OORBF_model'+str(i)+'.sav'
    joblib.dump(rbf,filename)

    viz = plot_roc_curve(rbf,x_test,y_test,name='RBF SVM model {}'.format(i), ax=ax)
    interp_tpr = np.interp(mean_fpr, viz.fpr, viz.tpr)
    interp_tpr[0] = 0.0
    tprs.append(interp_tpr)
    aucs.append(viz.roc_auc)
    i+=1

loaded_model = joblib.load('savemodel/OORBF_model5.sav')
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L195-L218



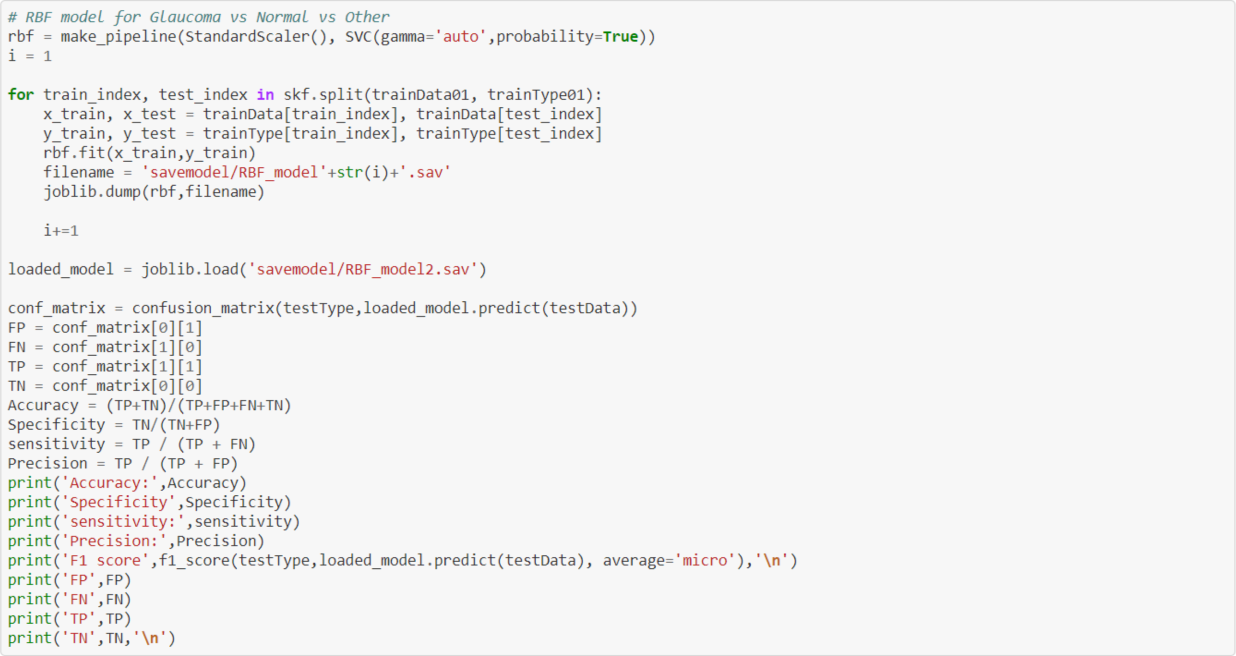
```
conf_matrix = confusion_matrix(testType12,loaded_model.predict(testData12))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType12,loaded_model.predict(testData12), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')

ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', alpha=.8)

ax.set(xlim=[-0.05, 1.05], ylim=[-0.05, 1.05],title="Other vs Non-Other")
ax.legend(loc="lower right")
plt.show()
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L221-L233



```python
# RBF model for Glaucoma vs Normal vs Other
cdr = np.reshape(dataAll.CDr.values,(-1,1))
cdd = np.reshape(dataAll.CDd.values,(-1,1))
data = np.hstack([cdr,cdd])
eye = dataAll.drop(['CDr','CDd'],axis=1).eye.values

trainData,testData,trainType,testType = train_test_split(data, eye, test_size=0.2 , random_state=1)
skf = StratifiedKFold(n_splits=5)

plt.scatter(data[:,0],data[:,1], c=eye)
plt.xlabel('CDR')
plt.ylabel('y')
print(data)
```

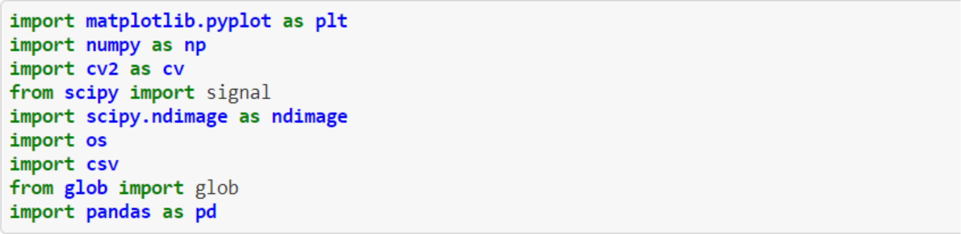
https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/MachinemodelRBF.py#L236-L268



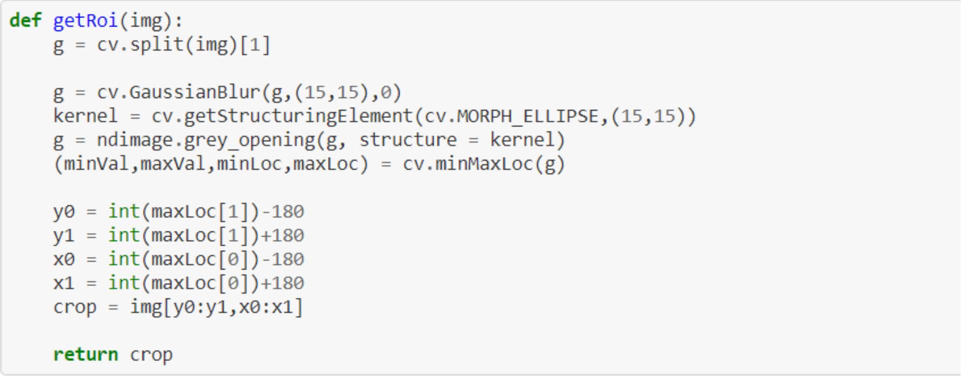
```python
# RBF model for Glaucoma vs Normal vs Other
rbf = make_pipeline(StandardScaler(), SVC(gamma='auto',probability=True))
i = 1

for train_index, test_index in skf.split(trainData01, trainType01):
    x_train, x_test = trainData[train_index], trainData[test_index]
    y_train, y_test = trainType[train_index], trainType[test_index]
    rbf.fit(x_train,y_train)
    filename = 'savemodel/RBF_model'+str(i)+'.sav'
    joblib.dump(rbf,filename)

    i+=1

loaded_model = joblib.load('savemodel/RBF_model2.sav')

conf_matrix = confusion_matrix(testType,loaded_model.predict(testData))
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
TN = conf_matrix[0][0]
Accuracy = (TP+TN)/(TP+FP+FN+TN)
Specificity = TN/(TN+FP)
sensitivity = TP / (TP + FN)
Precision = TP / (TP + FP)
print('Accuracy:',Accuracy)
print('Specificity',Specificity)
print('sensitivity:',sensitivity)
print('Precision:',Precision)
print('F1 score',f1_score(testType,loaded_model.predict(testData), average='micro'),'\n')
print('FP',FP)
print('FN',FN)
print('TP',TP)
print('TN',TN,'\n')
```

Extract Feature

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L11-L25

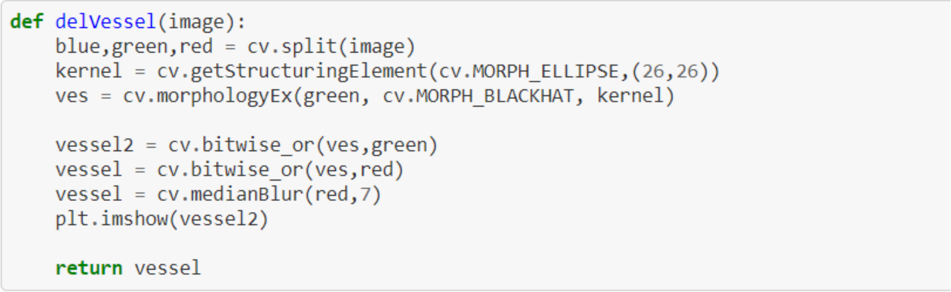


```
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv
from scipy import signal
import scipy.ndimage as ndimage
import os
import csv
from glob import glob
import pandas as pd
```



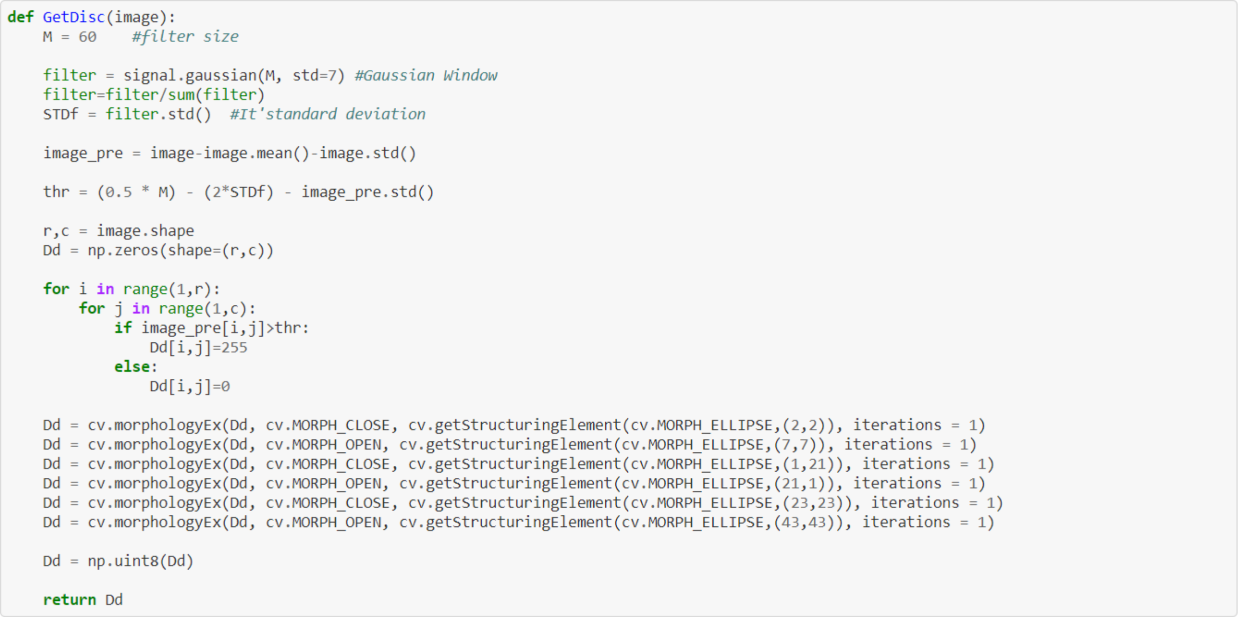
```
def getRoi(img):
    g = cv.split(img)[1]

    g = cv.GaussianBlur(g,(15,15),0)
    kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE,(15,15))
    g = ndimage.grey_opening(g, structure = kernel)
    (minVal,maxVal,minLoc,maxLoc) = cv.minMaxLoc(g)

    y0 = int(maxLoc[1])-180
    y1 = int(maxLoc[1])+180
    x0 = int(maxLoc[0])-180
    x1 = int(maxLoc[0])+180
    crop = img[y0:y1,x0:x1]

    return crop
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L27-L37



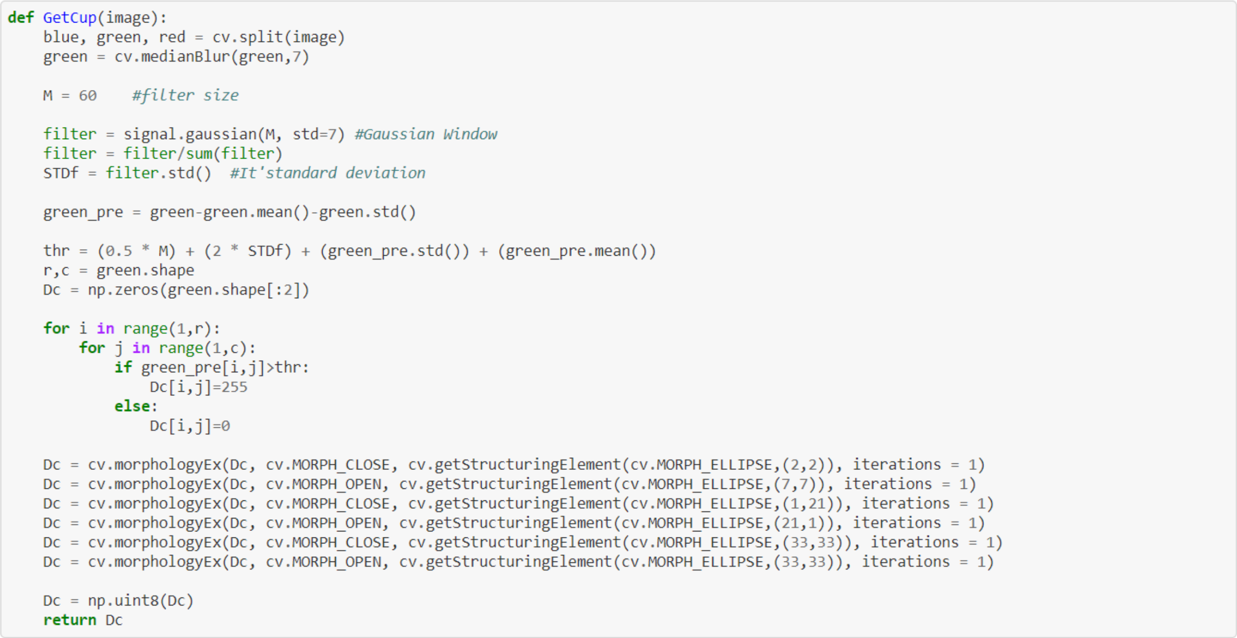
```
def delVessel(image):
    blue,green,red = cv.split(image)
    kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE,(26,26))
    ves = cv.morphologyEx(green, cv.MORPH_BLACKHAT, kernel)

    vessel2 = cv.bitwise_or(ves,green)
    vessel = cv.bitwise_or(ves,red)
    vessel = cv.medianBlur(red,7)
    plt.imshow(vessel2)

    return vessel
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L39-L69



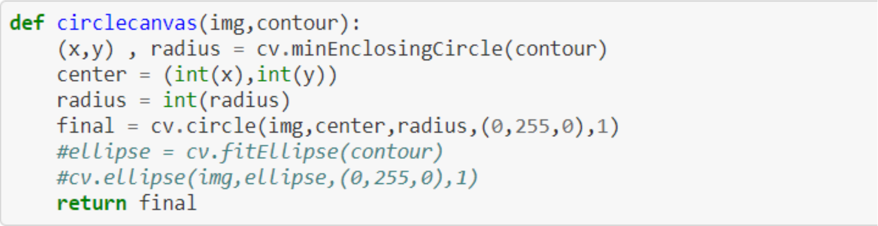
```python
def GetDisc(image):
    M = 60    #filter size

    filter = signal.gaussian(M, std=7) #Gaussian Window
    filter=filter/sum(filter)
    STDf = filter.std()  #It'standard deviation

    image_pre = image-image.mean()-image.std()

    thr = (0.5 * M) - (2*STDf) - image_pre.std()

    r,c = image.shape
    Dd = np.zeros(shape=(r,c))

    for i in range(1,r):
        for j in range(1,c):
            if image_pre[i,j]>thr:
                Dd[i,j]=255
            else:
                Dd[i,j]=0

    Dd = cv.morphologyEx(Dd, cv.MORPH_CLOSE, cv.getStructuringElement(cv.MORPH_ELLIPSE,(2,2)), iterations = 1)
    Dd = cv.morphologyEx(Dd, cv.MORPH_OPEN, cv.getStructuringElement(cv.MORPH_ELLIPSE,(7,7)), iterations = 1)
    Dd = cv.morphologyEx(Dd, cv.MORPH_CLOSE, cv.getStructuringElement(cv.MORPH_ELLIPSE,(1,21)), iterations = 1)
    Dd = cv.morphologyEx(Dd, cv.MORPH_OPEN, cv.getStructuringElement(cv.MORPH_ELLIPSE,(21,1)), iterations = 1)
    Dd = cv.morphologyEx(Dd, cv.MORPH_CLOSE, cv.getStructuringElement(cv.MORPH_ELLIPSE,(23,23)), iterations = 1)
    Dd = cv.morphologyEx(Dd, cv.MORPH_OPEN, cv.getStructuringElement(cv.MORPH_ELLIPSE,(43,43)), iterations = 1)

    Dd = np.uint8(Dd)

    return Dd
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L71-L102



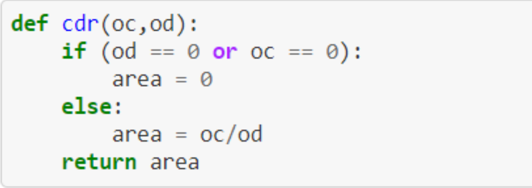
```python
def GetCup(image):
    blue, green, red = cv.split(image)
    green = cv.medianBlur(green,7)

    M = 60    #filter size

    filter = signal.gaussian(M, std=7) #Gaussian Window
    filter = filter/sum(filter)
    STDf = filter.std()  #It'standard deviation

    green_pre = green-green.mean()-green.std()

    thr = (0.5 * M) + (2 * STDf) + (green_pre.std()) + (green_pre.mean())
    r,c = green.shape
    Dc = np.zeros(green.shape[:2])

    for i in range(1,r):
        for j in range(1,c):
            if green_pre[i,j]>thr:
                Dc[i,j]=255
            else:
                Dc[i,j]=0

    Dc = cv.morphologyEx(Dc, cv.MORPH_CLOSE, cv.getStructuringElement(cv.MORPH_ELLIPSE,(2,2)), iterations = 1)
    Dc = cv.morphologyEx(Dc, cv.MORPH_OPEN, cv.getStructuringElement(cv.MORPH_ELLIPSE,(7,7)), iterations = 1)
    Dc = cv.morphologyEx(Dc, cv.MORPH_CLOSE, cv.getStructuringElement(cv.MORPH_ELLIPSE,(1,21)), iterations = 1)
    Dc = cv.morphologyEx(Dc, cv.MORPH_OPEN, cv.getStructuringElement(cv.MORPH_ELLIPSE,(21,1)), iterations = 1)
    Dc = cv.morphologyEx(Dc, cv.MORPH_CLOSE, cv.getStructuringElement(cv.MORPH_ELLIPSE,(33,33)), iterations = 1)
    Dc = cv.morphologyEx(Dc, cv.MORPH_OPEN, cv.getStructuringElement(cv.MORPH_ELLIPSE,(33,33)), iterations = 1)

    Dc = np.uint8(Dc)
    return Dc
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L104-L110



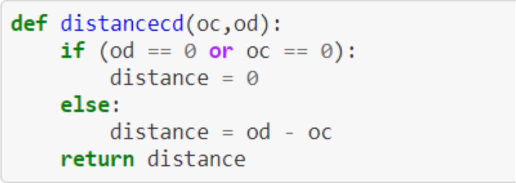
```
def circlecanvas(img,contour):
    (x,y) , radius = cv.minEnclosingCircle(contour)
    center = (int(x),int(y))
    radius = int(radius)
    final = cv.circle(img,center,radius,(0,255,0),1)
    #ellipse = cv.fitEllipse(contour)
    #cv.ellipse(img,ellipse,(0,255,0),1)
    return final
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L112-L117



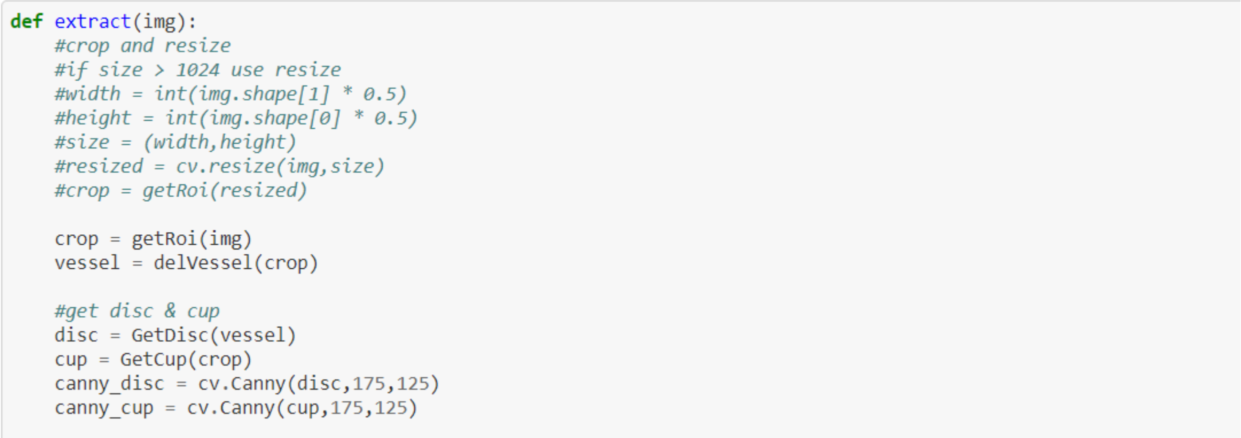
```
def cdr(oc,od):
    if (od == 0 or oc == 0):
        area = 0
    else:
        area = oc/od
    return area
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L119-L124



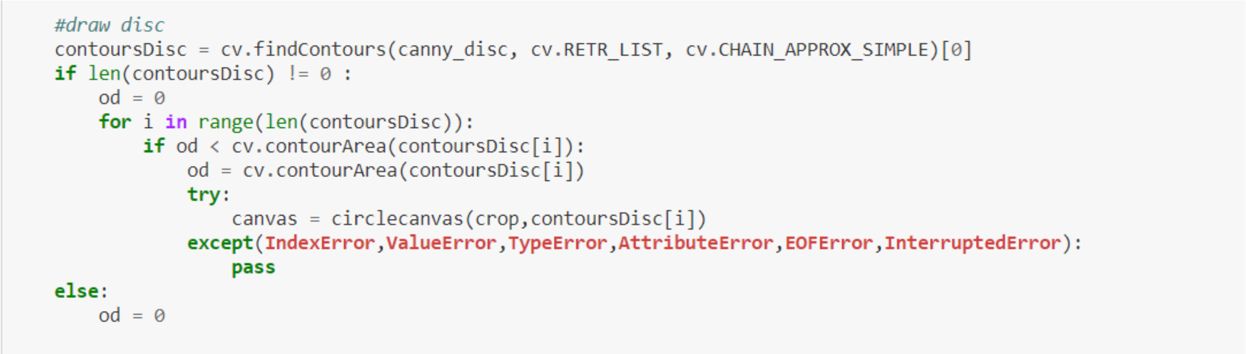
```
def distancecd(oc,od):
    if (od == 0 or oc == 0):
        distance = 0
    else:
        distance = od - oc
    return distance
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L126-L142



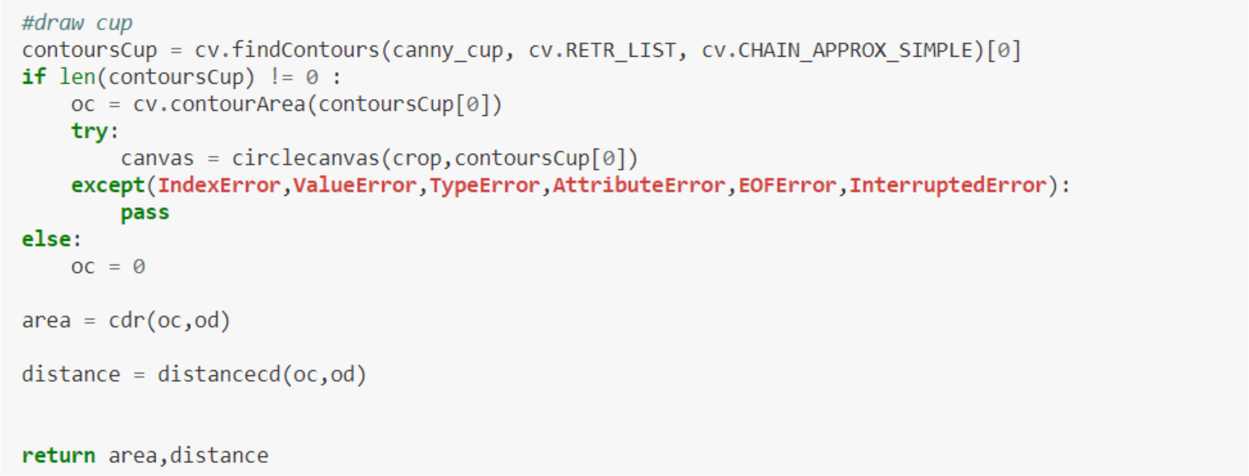
```
def extract(img):
    #crop and resize
    #if size > 1024 use resize
    #width = int(img.shape[1] * 0.5)
    #height = int(img.shape[0] * 0.5)
    #size = (width,height)
    #resized = cv.resize(img,size)
    #crop = getRoi(resized)

    crop = getRoi(img)
    vessel = delVessel(crop)

    #get disc & cup
    disc = GetDisc(vessel)
    cup = GetCup(crop)
    canny_disc = cv.Canny(disc,175,125)
    canny_cup = cv.Canny(cup,175,125)
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L144-L174



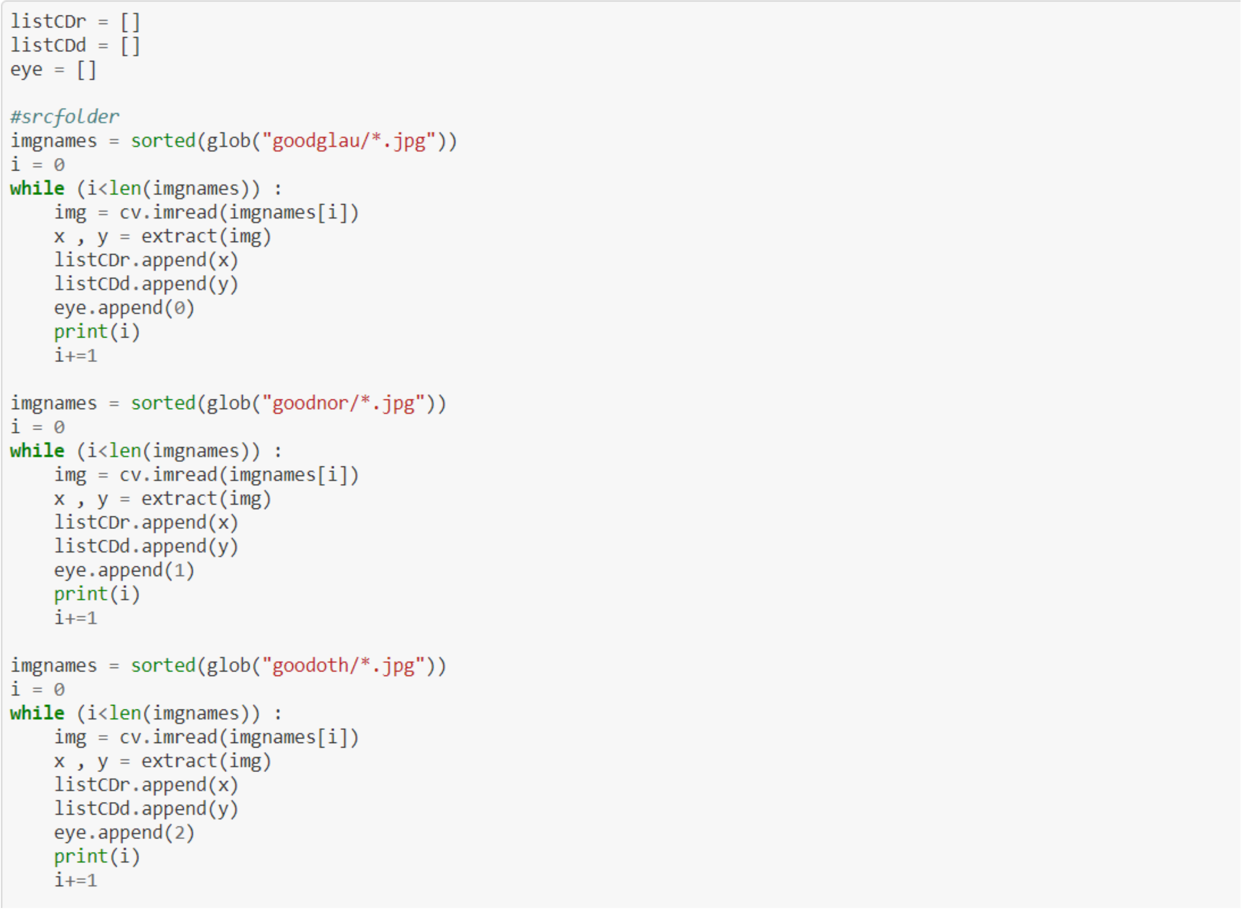
```
#draw disc
contoursDisc = cv.findContours(canny_disc, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)[0]
if len(contoursDisc) != 0 :
    od = 0
    for i in range(len(contoursDisc)):
        if od < cv.contourArea(contoursDisc[i]):
            od = cv.contourArea(contoursDisc[i])
            try:
                canvas = circlecanvas(crop,contoursDisc[i])
            except(IndexError,ValueError,TypeError,AttributeError,EOFError,InterruptedError):
                pass
else:
    od = 0
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L158-L174

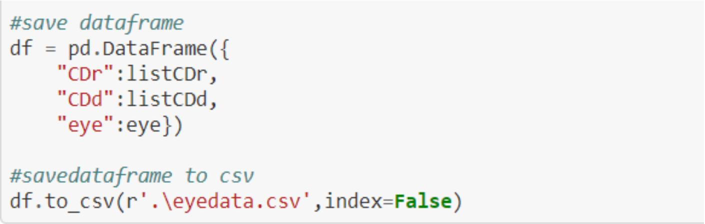


```
#draw cup
contoursCup = cv.findContours(canny_cup, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)[0]
if len(contoursCup) != 0 :
    oc = cv.contourArea(contoursCup[0])
    try:
        canvas = circlecanvas(crop,contoursCup[0])
    except(IndexError,ValueError,TypeError,AttributeError,EOFError,InterruptedError):
        pass
else:
    oc = 0

area = cdr(oc,od)

distance = distancecd(oc,od)


return area,distance
```

https://github.com/gringgyy/cn240/blob/dbd4ff9232935187971b723b6be7e4161dd5382c/machineLearning/ExtractFeature.py#L177-L223



```
listCDr = []
listCDd = []
eye = []

#srcfolder
imgnames = sorted(glob("goodglau/*.jpg"))
i = 0
while (i<len(imgnames)) :
    img = cv.imread(imgnames[i])
    x , y = extract(img)
    listCDr.append(x)
    listCDd.append(y)
    eye.append(0)
    print(i)
    i+=1

imgnames = sorted(glob("goodnor/*.jpg"))
i = 0
while (i<len(imgnames)) :
    img = cv.imread(imgnames[i])
    x , y = extract(img)
    listCDr.append(x)
    listCDd.append(y)
    eye.append(1)
    print(i)
    i+=1

imgnames = sorted(glob("goodoth/*.jpg"))
i = 0
while (i<len(imgnames)) :
    img = cv.imread(imgnames[i])
    x , y = extract(img)
    listCDr.append(x)
    listCDd.append(y)
    eye.append(2)
    print(i)
    i+=1
```



```
#save dataframe
df = pd.DataFrame({
    "CDr":listCDr,
    "CDd":listCDd,
    "eye":eye})

#savedataframe to csv
df.to_csv(r'.\eyedata.csv',index=False)
```