Hyperparameters of the models

```
#LogisticRegression
model_log=LogisticRegression(C=0.1)
model_log.fit(X_train,y_train)
pred_log=model_log.predict(X_test)
score_log=model_log.score(X_test,y_test)
score_log_train=model_log.score(X_train,y_train)
pred_log_train=model_log.predict(X_train)
print("log_test:",score_log,

#LinearDiscriminantAnalysis
model_lr=LinearDiscriminantAnalysis()
model_lr.fit(X_train,y_train)
pred_lr=model_lr.predict(X_test)
score_lr=model_lr.score(X_test,y_test)
score_lr_train=model_lr.score(X_train,y_train)
pred_lr_train=model_lr.predict(X_train)
print("lr_test:",score_lr,

#KNeighbors
model_knn=KNeighborsClassifier(n_neighbors=10)
model_knn.fit(X_train,y_train)
pred_knn=model_knn.predict(X_test)
score_knn=model_knn.score(X_test,y_test)
score_knn_train=model_knn.score(X_train,y_train)
pred_knn_train=model_knn.predict(X_train)
print("knn_test:",score_knn,

#DecisionTree
model_dtc=DecisionTreeClassifier(random_state=77)
model_dtc.fit(X_train,y_train)
pred_dtc=model_dtc.predict(X_test)
score_dtc=model_dtc.score(X_test,y_test)
score_dtc_train=model_dtc.score(X_train,y_train)
pred_dtc_train=model_dtc.predict(X_train)
print("dtc_test:",score_dtc,

#SVM
Cs=np.logspace(-1,3,10,base=2)#Adjust parameters
gammas=np.logspace(-4,1,50,base=2)
param_grid=dict(C=Cs,gamma=gammas)
grid=GridSearchCV(svm.SVC(kernel="rbf"),param_grid=param_grid,cv=10).fit(X,y)
```

```
print(grid.best_params_)
C=grid.best_params_["C"]
gamma=grid.best_params_["gamma"]
model_svm=svm.SVC(kernel="rbf",C=C,gamma=gamma,
random_state=77,probability=True)
model_svm.fit(X_train,y_train)
pred_svm=model_svm.predict(X_test)
score_svm=model_svm.score(X_test,y_test)
score_svm_train=model_svm.score(X_train,y_train)
pred_svm_train=model_svm.predict(X_train)
print("svm_test:",score_svm,

#ExtremeGradientBoosting
model_xgb=XGBClassifier()
model_xgb.fit(X_train, y_train)
pred_xgb=model_xgb.predict(X_test)
score_xgb=model_xgb.score(X_test,y_test)
score_xgb_train=model_xgb.score(X_train,y_train)
pred_xgb_train=model_xgb.predict(X_train)
print("xgb_test:",score_xgb,

#RandomForest
model_rf=RandomForestClassifier(n_estimators=20)
model_rf.fit(X_train,y_train)
pred_rf=model_rf.predict(X_test)
score_rf=model_rf.score(X_test,y_test)
score_rf_train=model_rf.score(X_train,y_train)
pred_rf_train=model_rf.predict(X_train)
print("rf_test:",score_rf,

#Lightgbm
model_lgb=lgb.LGBMClassifier()
model_lgb.fit(X_train,y_train)
pred_lgb=model_lgb.predict(X_test)
score_lgb=model_lgb.score(X_test,y_test)
score_lgb_train=model_lgb.score(X_train,y_train)
pred_lgb_train=model_lgb.predict(X_train)
print("lgb_test:",score_lgb,

#NeuralNetwork
model_mlp=MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(5,5,2), random_state=1)
model_mlp.fit(X_train,y_train) #fitting
pred_mlp=model_mlp.predict(X_test)
```

```
score_mlp=model_mlp.score(X_test,y_test,sample_weight=None)
pred_mlp_train=model_mlp.predict(X_train)
score_mlp_train=model_mlp.score(X_train,y_train,sample_weight=None)
print("test:",score_mlp,
#GradientBoosting
model_gbc=GradientBoostingClassifier(random_state=123)
model_gbc.fit(X_train,y_train)
pred_gbc=model_gbc.predict(X_test)
score_gbc=model_gbc.score(X_test,y_test)
score_gbc_train=model_gbc.score(X_train,y_train)
pred_gbc_train=model_gbc.predict(X_train)
print("gbc_test:",score_gbc,

#Adaboost
model_adb=AdaBoostClassifier(DecisionTreeClassifier(max_depth=3),n_estimators=
10)#AdaBoost
model_adb.fit(X_train, y_train)
pred_adb=model_adb.predict(X_test)
score_adb=model_adb.score(X_test,y_test)
score_adb_train=model_adb.score(X_train,y_train)
pred_adb_train=model_adb.predict(X_train)
print("adb_test:",score_adb,
```