```
from google.colab import drive
drive.mount('/content/gdrive')
```

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

```
# programmer - Sophia Quinton
# date - 12-22-21
# class - DSC -540
# assignment - Assignment 8

#acquire data
#(Pedregosa et. al., 2011) (Harris et. al., 2020)
import pandas as pd
import numpy as np

def normalize(column):
  # (Slave, 2018) (cat, 2015)
  x = column.replace(np.nan, 0)
  x = np.asarray(x, dtype=np.float64)
  x -= np.mean(x)
  x/=np.std(x)
  return x

features = pd.read_csv("/content/gdrive/MyDrive/data/feature_extract.csv",
sep=",", index_col=0)
features2 = features.iloc[: , :-1]
# (Varun, nd)

new_frame = pd.DataFrame()
for column in features2:
  normalized_column = normalize(features2[column])
  new_frame = pd.concat([new_frame, pd.DataFrame(normalized_column)], axis =
1)

new_frame.columns = ['meanx', 'meany', 'meanz', 'standardx', 'standardy',
       'standardz', 'minimumx', 'minimumy', 'minimumz', 'maximumx',
'maximumy',
       'maximumz', 'variancex', 'variancey', 'variancez', 'medianx',
'mediany',
       'medianz', 'skewnessx', 'skewnessy', 'skewnessz', 'percent_25x',
       'percent_25y', 'percent_25z', 'percent_75x', 'percent_75y',
       'percent_75z', 'kurtosisx', 'kurtosisy', 'kurtosisz', 'spectralE_x',
       'spectralE_y', 'spectralE_z', 'zero_crossx', 'zero_crossy',
       'zero_crossz', 'dominant_freqx', 'dominant_freqy', 'dominant_freqz',
       'corxy', 'corxz', 'coryz']

#classic
from sklearn.model_selection import train_test_split
y = features["activity"]
X = new_frame[["minimumx", "minimumy", "minimumz", "maximumy", "maximumz",
"meanx",
  "variancez", "standardx", "standardy", "standardz", "medianx",
"percent_25x",
  "percent_75x", "percent_25z", "zero_crossx", "zero_crossz",
"dominant_freqx"]]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=25)
X_train.reset_index(level=0, inplace=True)
X_test.reset_index(level=0, inplace=True)
y_train.reset_index(drop = True, inplace=True)


#bagging decision (Brownlee, 2021)
#(Pedregosa et. al., 2011)
from sklearn.ensemble import BaggingClassifier
bagging_decision = BaggingClassifier()
bagging_model = bagging_decision.fit(X_train, y_train)
ypred = bagging_model.predict(X_test)

#random forest
#(Pedregosa et. al., 2011)
total_features = np.sqrt(16)

from sklearn.ensemble import RandomForestClassifier
rfytrain = np.ravel(y_train)
rf1 = RandomForestClassifier(n_estimators = 20, criterion="gini",
max_features = "sqrt").fit(X_train,rfytrain)
rfpred1 = rf1.predict(X_test)


#boosting tree
from sklearn.ensemble import AdaBoostClassifier
Ada_class = AdaBoostClassifier(n_estimators=100)
Ada_model = Ada_class.fit(X_train, y_train)
y_predAda = Ada_model.predict(X_test)

#(Pedregosa et. al., 2011)
from sklearn.ensemble import VotingClassifier
Voting_method = VotingClassifier(estimators=[('bag', bagging_decision),
('forest', rf1), ('boost', Ada_class)],
                                 voting='hard')
voting_model = Voting_method.fit(X_train, y_train)
y_pred_vote = voting_model.predict(X_test)

## binary decision tree
from sklearn import tree
BDT_class = tree.DecisionTreeClassifier(max_depth = 20)
BDT_model = BDT_class.fit(X_train, y_train)
BDT_pred = BDT_model.predict(X_test)

# k nearest neighbor
from sklearn.neighbors import KNeighborsClassifier
```

```
Kmeans_model = KNeighborsClassifier(n_neighbors=7)
model_k = Kmeans_model.fit(X_train, y_train)
label = model_k.predict(X_test)

# SVM
from sklearn import svm
SVM_class = svm.SVC()
SVM_model = SVM_class.fit(X_train, y_train)
SVM_pred = SVM_model.predict(X_test)

from sklearn.neural_network import MLPClassifier
nn_class = MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(5,2), max_iter=1000)
nn_model = nn_class.fit(X_train, y_train)
nn_pred = nn_model.predict(X_test)

##WMV for custom
Voting_method2 = VotingClassifier(estimators=[('BDT', BDT_model), ('KNN',
Kmeans_model), ('SVM', SVM_model), ("ANN", nn_model)],
                                 voting='hard')
voting_model2 = Voting_method2.fit(X_train, y_train)
y_pred_vote2 = voting_model2.predict(X_test)

from sklearn.metrics import f1_score
bag_score = f1_score(y_test, ypred, average=None)
rf_score = f1_score(y_test, rfpred1, average=None)
boost_score = f1_score(y_test, y_predAda, average=None)

BDT_score = f1_score(y_test, BDT_pred, average=None)
KNN_score = f1_score(y_test, label, average=None)
SVM_score = f1_score(y_test, SVM_pred, average=None)
ANN_score = f1_score(y_test, nn_pred, average=None)

WVM = f1_score(y_test, y_pred_vote, average=None)
WVM2 = f1_score(y_test, y_pred_vote2, average=None)

single_standard = pd.concat([pd.DataFrame(bag_score), pd.DataFrame(rf_score),
pd.DataFrame(boost_score)], axis=1)
single_standard.columns = ["bagging", "random forest", "boost tree"]
single_standard.to_csv("/content/gdrive/MyDrive/data/single_standards.csv",
header=True)

single_custom = pd.concat([pd.DataFrame(BDT_score), pd.DataFrame(KNN_score),
                           pd.DataFrame(SVM_score), pd.DataFrame(ANN_score)],
axis=1)
single_custom.columns = ["BDT", "KNN", "SVM", "ANN"]
single_custom.to_csv("/content/gdrive/MyDrive/data/single_customs.csv",
header=True)
```

```
WVM_frame = pd.concat([pd.DataFrame(WVM), pd.DataFrame(WVM2)], axis=1)
WVM_frame.columns = ["WVM - Standard", "WVM - Custom"]
WVM_frame.to_csv("/content/gdrive/MyDrive/data/WVM.csv", header=True)
```