AHSAN JAVED
BM-22075

NEUROCOMPUTATION LAB
NED UNIVERSITY OF ENGINEERING & TECHNOLOGY
KARACHI

MACHINE LEARNING INTERNSHIP
Task 2:

Download the data for Task 2 from the link below:
https://drive.google.com/drive/folders/1iJTyZogi05PnzoxJLGA2O9nUHAFbWwF-?usp=sharing

Instructions regarding the data provided:
Similar to the previous task, you have been provided with subject data for Stages 0-4. However, now within each of those stages are two conditions, EO and EC. Make sure you train and test the models separately for the EO and EC conditions.
Again, as with the last task, there are 6 features for each condition at each stage named 'F1', 'F2', 'F3', 'F4', 'F5' and 'F6' having the size [number of subjects at the stage] x 38 channels. These are available as .mat and .csv files both. Remember to keep the 38 channels separate from one another. You may treat them as 38 separate attributes.

Task:
You are required to test and train models based on the following algorithms:

1. Random Forest
2. Multinomial Naive Bayes
3. LightGBM
4. Linear SVM
5. Non-Linear SVM

Create 5 models, all to classify Stage 0 and Stage 1. Test this model using the data from Stage 2,3 and 4. (Remember, first you need to train and test the model using all EO data, then do the same for all EC data). You need to note how many of Stage 2,3 and 4 subjects get classified as Stage 0 and Stage 1. Fill the tables provided below with your results, and add any additional tables or information after the given tables in your report.

For Stage 0 v 1, EO	Accuracy	Precision	Recall	F1 Score
Random Forest	0.89	0.86	0.93	0.89
Multinomial Naive Bayes	0.60	0.60	0.54	0.57

AHSAN JAVED
BM-22075

LightGBM	0.86	0.81	0.93	0.86
Linear SVM	0.78	0.70	0.95	0.81
Non-Linear SVM	0.73	0.64	1.0	0.78
For Stage 0 v 1, EC	Accuracy	Precision	Recall	F1 Score
Random Forest	0.91	0.88	0.95	0.91
Multinomial Naive Bayes	0.50	0.5	0.54	0.52
LightGBM	0.92	0.86	1.0	0.92
Linear SVM	0.83	0.76	0.97	0.85
Non-Linear SVM	0.77	0.71	0.91	0.8

	Testing Dataset		
	Stage 2 EO	Stage 3 EO	Stage 4 EO
Random Forest			
Classified as Stage 0	85	42	21
Classified as Stage 1	71	36	15
Multinomial Naive Bayes			
Classified as Stage 0	59	33	13
Classified as Stage 1	97	45	23
LightGBM			
Classified as Stage 0	85	47	22
Classified as Stage 1	71	31	14
Linear SVM			
Classified as Stage 0	52	29	10
Classified as Stage 1	104	49	26
Non-Linear SVM			
Classified as Stage 0	21	19	3
Classified as Stage 1	135	59	33

AHSAN JAVED
BM-22075

	Testing Dataset		
	Stage 2 EC	Stage 3 EC	Stage 4 EC
Random Forest			
Classified as Stage 0	94	60	23
Classified as Stage 1	62	18	19
Multinomial Naive Bayes			
Classified as Stage 0	65	34	17
Classified as Stage 1	91	44	25
LightGBM			
Classified as Stage 0	98	65	26
Classified as Stage 1	58	13	16
Linear SVM			
Classified as Stage 0	46	33	17
Classified as Stage 1	110	45	25
Non-Linear SVM			
Classified as Stage 0	36	37	6
Classified as Stage 1	120	41	36

AHSAN JAVED
BM-22075

TASK REPORT

1. IMPORT LIBRARIES

The code imports the following libraries:

- pandas (pd): for data manipulation
- numpy (np): for numerical operations
- os: for file system interaction
- sklearn.model_selection: for splitting data into training and testing sets
- seaborn (sns): for data visualization (likely not used in this script)
- imblearn.pipeline: for building pipelines with imbalanced data handling
- imblearn.over_sampling: for oversampling techniques (RandomOverSampler used)
- sklearn.ensemble: for ensemble learning methods (RandomForestClassifier)
- sklearn.naive_bayes: for naive bayes classifiers (MultinomialNB)
- lightgbm: for LightGBM classifiers (LGBMClassifier)
- sklearn.svm: for support vector machines (LinearSVC, SVC)
- sklearn.metrics: for calculating evaluation metrics (accuracy_score, precision_score, recall_score, f1_score)

2. DATA PATH DISCOVERY

The code iterates through a directory (C:\infusion pump\Task2) to find all CSV files. It then separates files based on whether they contain "EO" or "EC" in the filename, likely indicating Event Occurrence or patient Condition data. Finally, it further separates these files based on stages (Stage2, Stage3, Stage4) potentially indicating the severity of the event or condition.

3. DATA LOADING AND PREPROCESSING

- **Training Data:**
  - The code identifies files containing "Stage0" or "Stage1" which are likely considered training data for both EO and EC categories.
  - It reads each CSV file and assigns a label (0 for Stage0, 1 otherwise) based on the filename (assuming Stage1 is positive class).
  - It concatenates all training dataframes for EO and EC separately into EO_df and EC_df.
  - For both dataframes, it separates features (all columns except "label") and target variable ("label").
  - It applies RandomOverSampling to address potential class imbalance in the data.
- **Testing Data:**
  - The code splits the features (X) and target variable (y) for both EO and EC dataframes into training and testing sets using train_test_split while maintaining class distribution through stratification.

AHSAN JAVED
BM-22075

4. MODEL BUILDING AND EVALUATION

- **Machine Learning Models:**
  - The code defines several machine learning models using pipelines:
    - RandomForestClassifier
    - MultinomialNB
    - LGBMClassifier
    - LinearSVC
    - SVC (with RBF kernel)
- **Evaluation Function:**
  - A function check_model_metrics iterates through the defined pipelines, trains each model on the training data, and evaluates its performance on the testing data using the following metrics:
    - Accuracy
    - Precision
    - Recall
    - F1-score
  - The function prints the model name and the calculated metrics for each model.

5. STAGE CLASSIFICATION

- **Data Loading:**
  - Similar to training data, the code reads CSV files for each stage (2, 3, 4) within EO and EC categories.
  - It concatenates these dataframes into separate dataframes for each stage and category.
- **Classification Function:**
  - A function classify_stage_points iterates through the stages dictionary (containing dataframes for each stage and category) as well as the defined machine learning pipelines.
  - For each stage-category combination, it uses each model to predict the class labels for the corresponding data.
  - It then calculates the value counts (number of data points in each predicted class) and prints the model name along with the value counts for each stage-category combination.

AHSAN JAVED
BM-22075

THE CODE OF THE TASK IS BELOW IN DETAIL

```
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
import seaborn as sns
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import RandomOverSampler
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from lightgbm import LGBMClassifier
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score
file_paths = []
folder_file_paths = []

for root, dirs, files in os.walk('C:\infusion pump\Task2'):
    # Iterate over files in the current directory
    for file in files:
      if file.lower().endswith('.csv'):
    #     # Construct the full path to the file
        file_path = os.path.join(root, file)
        # Append the file path to the list
        folder_file_paths.append(file_path)
        folder_file_paths.sort()
    file_paths.append(folder_file_paths)
    folder_file_paths = []
file_paths=list(filter(None,file_paths))

EO = []
EC = []

for files in file_paths:
  for csv_file in files:
    if 'EO' in csv_file:
      EO.append(csv_file)
    else:
      EC.append(csv_file)
```

AHSAN JAVED
BM-22075

```
EO
```

```
['C:\\infusion pump\\Task2\\Stage0\\EO\\f1.csv',
 'C:\\infusion pump\\Task2\\Stage0\\EO\\f2.csv',
 'C:\\infusion pump\\Task2\\Stage0\\EO\\f3.csv',
 'C:\\infusion pump\\Task2\\Stage0\\EO\\f4.csv',
 'C:\\infusion pump\\Task2\\Stage0\\EO\\f5.csv',
 'C:\\infusion pump\\Task2\\Stage0\\EO\\f6.csv',
 'C:\\infusion pump\\Task2\\Stage1\\EO\\f1.csv',
 'C:\\infusion pump\\Task2\\Stage1\\EO\\f2.csv',
 'C:\\infusion pump\\Task2\\Stage1\\EO\\f3.csv',
 'C:\\infusion pump\\Task2\\Stage1\\EO\\f4.csv',
 'C:\\infusion pump\\Task2\\Stage1\\EO\\f5.csv',
 'C:\\infusion pump\\Task2\\Stage1\\EO\\f6.csv',
 'C:\\infusion pump\\Task2\\Stage2\\EO\\f1.csv',
 'C:\\infusion pump\\Task2\\Stage2\\EO\\f2.csv',
 'C:\\infusion pump\\Task2\\Stage2\\EO\\f3.csv',
 'C:\\infusion pump\\Task2\\Stage2\\EO\\f4.csv',
 'C:\\infusion pump\\Task2\\Stage2\\EO\\f5.csv',
 'C:\\infusion pump\\Task2\\Stage2\\EO\\f6.csv',
 'C:\\infusion pump\\Task2\\Stage3\\EO\\f1.csv',
 'C:\\infusion pump\\Task2\\Stage3\\EO\\f2.csv',
 'C:\\infusion pump\\Task2\\Stage3\\EO\\f3.csv',
 'C:\\infusion pump\\Task2\\Stage3\\EO\\f4.csv',
 'C:\\infusion pump\\Task2\\Stage3\\EO\\f5.csv',
 'C:\\infusion pump\\Task2\\Stage3\\EO\\f6.csv',
 'C:\\infusion pump\\Task2\\Stage4\\EO\\f1.csv',
 'C:\\infusion pump\\Task2\\Stage4\\EO\\f2.csv',
 'C:\\infusion pump\\Task2\\Stage4\\EO\\f3.csv',
 'C:\\infusion pump\\Task2\\Stage4\\EO\\f4.csv',
 'C:\\infusion pump\\Task2\\Stage4\\EO\\f5.csv',
 'C:\\infusion pump\\Task2\\Stage4\\EO\\f6.csv']
```

```
stage_2_eo = [features for features in EO if 'Stage2' in features]
stage_3_eo = [features for features in EO if 'Stage3' in features]
stage_4_eo = [features for features in EO if 'Stage4' in features]

stage_2_ec = [features for features in EC if 'Stage2' in features]
stage_3_ec = [features for features in EC if 'Stage3' in features]
stage_4_ec = [features for features in EC if 'Stage4' in features]

EO_train_data = [features for features in EO if 'Stage0' in features or 'Stage1' in features]
EC_train_data = [features for features in EC if 'Stage0' in features or 'Stage1' in features]

for items in EO_train_data:
  df = pd.read_csv(items)
  print(df.shape)
```

```
(30, 38)
(30, 38)
(30, 38)
(30, 38)
(30, 38)
(30, 38)
(11, 38)
(11, 38)
(11, 38)
(11, 38)
(11, 38)
(11, 38)
```

AHSAN JAVED
BM-22075

```
file_list = []
for csv_file in EO_train_data:
  if 'Stage0' in csv_file:
    temp = pd.read_csv(csv_file,header=None)
    temp['label'] = 0
  else:
    temp = pd.read_csv(csv_file,header=None)
    temp['label'] = 1
  file_list.append(temp)

EO_df = pd.concat(file_list,ignore_index=True,axis=0)

file_list = []
for csv_file in EC_train_data:
  if 'Stage0' in csv_file:
    temp = pd.read_csv(csv_file,header=None)
    temp['label'] = 0
  else:
    temp = pd.read_csv(csv_file,header=None)
    temp['label'] = 1
  file_list.append(temp)

EC_df = pd.concat(file_list,ignore_index=True,axis=0)
```

```
X = EO_df.drop(['label'],axis=1)
y = EO_df['label']
sampler = RandomOverSampler()
X,y= sampler.fit_resample(X,y)



x_train,x_test,y_train,y_test = train_test_split(X,y,stratify=y,random_state=42)



rf_classifier = Pipeline([('classifier',RandomForestClassifier(random_state=42))])

Mn_naive_classifier = Pipeline([('classifier',MultinomialNB())])

lgbm_classifier = Pipeline([('classifier',LGBMClassifier(random_state=42,verbosity=-1))])

lsvc_classifier = Pipeline([('classifier',LinearSVC(random_state=42,verbose=False))])

svc_classifier = Pipeline([('classifier',SVC(kernel='rbf',random_state=42))])

pipelines = [rf_classifier,Mn_naive_classifier,lgbm_classifier,lsvc_classifier,svc_classifier]



def check_model_metrics(model_pipelines):
  for model in model_pipelines:
    model.fit(x_train,y_train)
    print(model.named_steps['classifier'])
    print('accuracy',accuracy_score(y_test,model.predict(x_test)))
    print('precision',precision_score(y_test,model.predict(x_test)))
    print('recall score',recall_score(y_test,model.predict(x_test)))
    print('f1_score',f1_score(y_test,model.predict(x_test)))
    print('\n')
```

AHSAN JAVED
BM-22075

```
check_model_metrics(pipelines)
```

```
RandomForestClassifier(random_state=42)
accuracy 0.8924731182795699
precision 0.86
recall score 0.9347826086956522
f1_score 0.8958333333333334


MultinomialNB()
accuracy 0.6021505376344086
precision 0.6097560975609756
recall score 0.5434782608695652
f1_score 0.5747126436781609


LGBMClassifier(random_state=42, verbosity=-1)
accuracy 0.8602150537634409
precision 0.8113207547169812
recall score 0.9347826086956522
f1_score 0.8686868686868687


LinearSVC(random_state=42, verbose=False)
accuracy 0.7849462365591398
precision 0.7096774193548387
recall score 0.9565217391304348
f1_score 0.8148148148148148


SVC(random_state=42)
accuracy 0.7311827956989247
precision 0.647887323943662
recall score 1.0
f1_score 0.7863247863247863
```

AHSAN JAVED
BM-22075

```
temp = []
for files in stage_2_eo:
  temp.append(pd.concat([pd.read_csv(files,header=None)],ignore_index=True,axis=0))

stage_2_eo_df = pd.concat(temp,ignore_index=True,axis=0)

temp = []
for files in stage_3_eo:
  temp.append(pd.concat([pd.read_csv(files,header=None)],ignore_index=True,axis=0))

stage_3_eo_df = pd.concat(temp,ignore_index=True,axis=0)

temp = []
for files in stage_4_eo:
  temp.append(pd.concat([pd.read_csv(files,header=None)],ignore_index=True,axis=0))

stage_4_eo_df = pd.concat(temp,ignore_index=True,axis=0)

stages = {'Stage2 EO':stage_2_eo_df,'Stage3 EO':stage_3_eo_df,'Stage4 EO':stage_4_eo_df}
```

```
def classify_stage_points(model_pipelines,stages):
  for key,value in stages.items():
    print(key)
    for model in model_pipelines:
      unique_values,counts=np.unique(model.predict(value),return_counts=True)
      value_counts_dict = dict(zip(unique_values, counts))
      print(model.named_steps['classifier'],'\n',value_counts_dict)
    print('\n')
```

```
classify_stage_points(pipelines,stages)
```

AHSAN JAVED
BM-22075

```
classify_stage_points(pipelines,stages)
```

```
Stage2 EO
RandomForestClassifier(random_state=42)
 {0: 85, 1: 71}
MultinomialNB()
 {0: 59, 1: 97}
LGBMClassifier(random_state=42, verbosity=-1)
 {0: 85, 1: 71}
LinearSVC(random_state=42, verbose=False)
 {0: 52, 1: 104}
SVC(random_state=42)
 {0: 21, 1: 135}


Stage3 EO
RandomForestClassifier(random_state=42)
 {0: 42, 1: 36}
MultinomialNB()
 {0: 33, 1: 45}
LGBMClassifier(random_state=42, verbosity=-1)
 {0: 47, 1: 31}
LinearSVC(random_state=42, verbose=False)
 {0: 29, 1: 49}
SVC(random_state=42)
 {0: 19, 1: 59}


Stage4 EO
RandomForestClassifier(random_state=42)
 {0: 21, 1: 15}
MultinomialNB()
 {0: 13, 1: 23}
LGBMClassifier(random_state=42, verbosity=-1)
 {0: 22, 1: 14}
LinearSVC(random_state=42, verbose=False)
 {0: 10, 1: 26}
SVC(random_state=42)
 {0: 3, 1: 33}
```

```
X = EC_df.drop(['label'],axis=1)
y = EC_df['label']
sampler = RandomOverSampler()
X,y= sampler.fit_resample(X,y)
```

```
x_train,x_test,y_train,y_test = train_test_split(X,y,stratify=y,random_state=42)

check_model_metrics(pipelines)
```

```
RandomForestClassifier(random_state=42)
accuracy 0.9139784946236559
precision 0.88
recall score 0.9565217391304348
f1_score 0.9166666666666666
```

AHSAN JAVED
BM-22075

```
x_train,x_test,y_train,y_test = train_test_split(X,y,stratify=y,random_state=42)

check_model_metrics(pipelines)
```

```
RandomForestClassifier(random_state=42)
accuracy 0.9139784946236559
precision 0.88
recall score 0.9565217391304348
f1_score 0.9166666666666666


MultinomialNB()
accuracy 0.5053763440860215
precision 0.5
recall score 0.5434782608695652
f1_score 0.5208333333333334


LGBMClassifier(random_state=42, verbosity=-1)
accuracy 0.9247311827956989
precision 0.8679245283018868
recall score 1.0
f1_score 0.9292929292929293


LinearSVC(random_state=42, verbose=False)
accuracy 0.8387096774193549
precision 0.7627118644067796
recall score 0.9782608695652174
f1_score 0.8571428571428571


SVC(random_state=42)
accuracy 0.7741935483870968
precision 0.711864406779661
recall score 0.9130434782608695
f1_score 0.8
```

```
temp = []
for files in stage_2_ec:
  temp.append(pd.concat([pd.read_csv(files,header=None)],ignore_index=True,axis=0))

stage_2_ec_df = pd.concat(temp,ignore_index=True,axis=0)

temp = []
for files in stage_3_ec:
  temp.append(pd.concat([pd.read_csv(files,header=None)],ignore_index=True,axis=0))

stage_3_ec_df = pd.concat(temp,ignore_index=True,axis=0)

temp = []
for files in stage_4_ec:
  temp.append(pd.concat([pd.read_csv(files,header=None)],ignore_index=True,axis=0))

stage_4_ec_df = pd.concat(temp,ignore_index=True,axis=0)

stages = {'Stage 2 EC':stage_2_ec_df,'Stage 3 EC':stage_3_ec_df,'Stage 4 EC':stage_4_ec_df}
```

AHSAN JAVED
BM-22075

```
classify_stage_points(pipelines,stages)
```

```
Stage 2 EC
RandomForestClassifier(random_state=42)
 {0: 94, 1: 62}
MultinomialNB()
 {0: 65, 1: 91}
LGBMClassifier(random_state=42, verbosity=-1)
 {0: 98, 1: 58}
LinearSVC(random_state=42, verbose=False)
 {0: 46, 1: 110}
SVC(random_state=42)
 {0: 36, 1: 120}


Stage 3 EC
RandomForestClassifier(random_state=42)
 {0: 60, 1: 18}
MultinomialNB()
 {0: 34, 1: 44}
LGBMClassifier(random_state=42, verbosity=-1)
 {0: 65, 1: 13}
LinearSVC(random_state=42, verbose=False)
 {0: 33, 1: 45}
SVC(random_state=42)
 {0: 37, 1: 41}


Stage 4 EC
RandomForestClassifier(random_state=42)
 {0: 23, 1: 19}
MultinomialNB()
 {0: 17, 1: 25}
LGBMClassifier(random_state=42, verbosity=-1)
 {0: 26, 1: 16}
LinearSVC(random_state=42, verbose=False)
 {0: 17, 1: 25}
SVC(random_state=42)
 {0: 6, 1: 36}
```