Importing libraries and data

CEN481 - INTRODUCTION TO DATA MINING

SUPPORT VECTOR MACHINE(SVM)

ANIL ALKIŞ

2019556006

```
[17] from sklearn.model_selection import train_test_split, cross_val_score ,StratifiedKFold,GridSearchCV
     from sklearn.preprocessing import StandardScaler
     from sklearn.svm import SVC
     from sklearn import preprocessing
     from sklearn.pipeline import Pipeline
     from sklearn.metrics import accuracy_score, classification_report , confusion_matrix, ConfusionMatrixDisplay
     import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     import time
```

```
[18] dataset = pd.read_csv('/content/AcousticFeatures.csv')
```

Split

```
X = dataset[['_RMSenergy_Mean', '_Lowenergy_Mean', '_Fluctuation_Mean',
       '_Tempo_Mean', '_MFCC_Mean_1', '_MFCC_Mean_2', '_MFCC_Mean_3',
       '_MFCC_Mean_4', '_MFCC_Mean_5', '_MFCC_Mean_6', '_MFCC_Mean_7',
       '_MFCC_Mean_8', '_MFCC_Mean_9', '_MFCC_Mean_10', '_MFCC_Mean_11',
       '_MFCC_Mean_12', '_MFCC_Mean_13', '_Roughness_Mean', '_Roughness_Slope',
       '_Zero-crossingrate_Mean', '_AttackTime_Mean', '_AttackTime_Slope',
       '_Rolloff_Mean', '_Eventdensity_Mean', '_Pulseclarity_Mean',
       '_Brightness_Mean', '_Spectralcentroid_Mean', '_Spectralspread_Mean',
       '_Spectralskewness_Mean', '_Spectralkurtosis_Mean',
       '_Spectralflatness_Mean', '_EntropyofSpectrum_Mean',
       '_Chromagram_Mean_1', '_Chromagram_Mean_2', '_Chromagram_Mean_3',
       '_Chromagram_Mean_4', '_Chromagram_Mean_5', '_Chromagram_Mean_6',
       '_Chromagram_Mean_7', '_Chromagram_Mean_8', '_Chromagram_Mean_9',
       '_Chromagram_Mean_10', '_Chromagram_Mean_11', '_Chromagram_Mean_12',
       '_HarmonicChangeDetectionFunction_Mean',
       '_HarmonicChangeDetectionFunction_Std',
       '_HarmonicChangeDetectionFunction_Slope',
       '_HarmonicChangeDetectionFunction_PeriodFreq',
       '_HarmonicChangeDetectionFunction_PeriodAmp',
       '_HarmonicChangeDetectionFunction_PeriodEntropy']]
y = dataset['Class']
```

```
[20] dataset.isna().sum()
```

```
Class                                          0
_RMSenergy_Mean                                0
_Lowenergy_Mean                                0
_Fluctuation_Mean                              0
_Tempo_Mean                                    0
_MFCC_Mean_1                                   0
_MFCC_Mean_2                                   0
_MFCC_Mean_3                                   0
_MFCC_Mean_4                                   0
_MFCC_Mean_5                                   0
_MFCC_Mean_6                                   0
_MFCC_Mean_7                                   0
_MFCC_Mean_8                                   0
_MFCC_Mean_9                                   0
_MFCC_Mean_10                                  0
_MFCC_Mean_11                                  0
_MFCC_Mean_12                                  0
_MFCC_Mean_13                                  0
_Roughness_Mean                                0
_Roughness_Slope                               0
_Zero-crossingrate_Mean                        0
_AttackTime_Mean                               0
_AttackTime_Slope                              0
_Rolloff_Mean                                  0
_Eventdensity_Mean                             0
_Pulseclarity_Mean                             0
_Brightness_Mean                               0
_Spectralcentroid_Mean                         0
_Spectralspread_Mean                           0
_Spectralskewness_Mean                         0
_Spectralkurtosis_Mean                         0
_Spectralflatness_Mean                         0
_EntropyofSpectrum_Mean                        0
_Chromagram_Mean_1                             0
_Chromagram_Mean_2                             0
_Chromagram_Mean_3                             0
_Chromagram_Mean_4                             0
_Chromagram_Mean_5                             0
_Chromagram_Mean_6                             0
_Chromagram_Mean_7                             0
_Chromagram_Mean_8                             0
_Chromagram_Mean_9                             0
_Chromagram_Mean_10                            0
_Chromagram_Mean_11                            0
_Chromagram_Mean_12                            0
_HarmonicChangeDetectionFunction_Mean          0
_HarmonicChangeDetectionFunction_Std           0
_HarmonicChangeDetectionFunction_Slope         0
_HarmonicChangeDetectionFunction_PeriodFreq    0
_HarmonicChangeDetectionFunction_PeriodAmp     0
_HarmonicChangeDetectionFunction_PeriodEntropy 0
dtype: int64
```

no missing values.

```
[21] song_types = dataset["Class"].value_counts()
     song_types_df = pd.DataFrame(song_types)
     song_types_df = song_types.reset_index(level = 0)
     song_types_df
```

	index	Class
0	relax	100
1	happy	100
2	sad	100
3	angry	100

```
dataset.describe().T
```

	count	mean	std	min	25%	50%	75%	max
_RMSenergy_Mean	400.0	0.134650	0.064368	0.010	0.08500	0.1280	0.17400	0.431
_Lowenergy_Mean	400.0	0.553605	0.050750	0.302	0.52300	0.5530	0.58325	0.703
_Fluctuation_Mean	400.0	7.145932	2.280145	3.580	5.85950	6.7340	7.82350	23.475
_Tempo_Mean	400.0	123.682020	34.234344	48.284	101.49025	120.1325	148.98625	195.026
_MFCC_Mean_1	400.0	2.456422	0.799262	0.323	1.94850	2.3895	2.86025	5.996
_MFCC_Mean_2	400.0	0.071890	0.537865	-3.484	-0.26275	0.0685	0.41325	1.937
_MFCC_Mean_3	400.0	0.488065	0.294607	-0.870	0.28125	0.4645	0.68600	1.622
_MFCC_Mean_4	400.0	0.030465	0.275839	-1.636	-0.11700	0.0445	0.19825	1.126
_MFCC_Mean_5	400.0	0.178897	0.195230	-0.494	0.06125	0.1810	0.28850	1.055
_MFCC_Mean_6	400.0	0.038307	0.203754	-0.916	-0.07825	0.0495	0.15125	0.799
_MFCC_Mean_7	400.0	0.059943	0.180982	-0.936	-0.04125	0.0720	0.17225	0.571
_MFCC_Mean_8	400.0	0.043467	0.165184	-0.744	-0.04925	0.0395	0.13000	0.728
_MFCC_Mean_9	400.0	0.023010	0.159239	-0.621	-0.07100	0.0165	0.12300	0.539
_MFCC_Mean_10	400.0	0.027793	0.152235	-0.544	-0.05925	0.0315	0.12600	0.510
_MFCC_Mean_11	400.0	0.028798	0.136156	-0.487	-0.04400	0.0370	0.11400	0.494
_MFCC_Mean_12	400.0	0.016667	0.128528	-0.418	-0.05600	0.0225	0.09450	0.355
_MFCC_Mean_13	400.0	0.024118	0.133470	-0.620	-0.04550	0.0390	0.10125	0.536
_Roughness_Mean	400.0	527.681365	521.218943	0.941	169.18875	367.5780	734.37250	3899.847
_Roughness_Slope	400.0	0.072038	0.174301	-0.525	-0.02700	0.0680	0.17400	0.584
_Zero-crossingrate_Mean	400.0	997.252315	524.895867	149.490	592.27500	893.4910	1303.49275	3147.907
_AttackTime_Mean	400.0	0.031305	0.016801	0.010	0.02300	0.0270	0.03300	0.165
_AttackTime_Slope	400.0	-0.002890	0.149920	-0.465	-0.09400	0.0075	0.08900	0.599
_Rolloff_Mean	400.0	5691.069637	2293.401839	887.151	3933.55275	5648.6280	7355.88625	11508.298
_Eventdensity_Mean	400.0	2.784820	1.326889	0.234	1.73700	2.7730	3.69250	7.952
_Pulseclarity_Mean	400.0	0.249387	0.155335	0.011	0.12775	0.2180	0.32725	0.856
_Brightness_Mean	400.0	0.434158	0.131517	0.053	0.35250	0.4480	0.52725	0.737
_Spectralcentroid_Mean	400.0	2581.167267	863.520318	606.524	1981.55775	2547.6780	3182.56975	5326.379
_Spectralspread_Mean	400.0	3082.394695	767.648035	814.817	2506.76850	3150.9490	3684.32525	4721.479
_Spectralskewness_Mean	400.0	1.870035	0.881635	0.390	1.32725	1.6870	2.18250	7.855
_Spectralkurtosis_Mean	400.0	7.348953	8.621386	1.930	3.88150	5.2160	7.84900	121.996
_Spectralflatness_Mean	400.0	0.048523	0.026492	0.006	0.02900	0.0470	0.06200	0.209
_EntropyofSpectrum_Mean	400.0	0.872607	0.037260	0.740	0.85300	0.8790	0.89900	0.942
_Chromagram_Mean_1	400.0	0.352560	0.323071	0.000	0.05700	0.2735	0.55125	1.000
_Chromagram_Mean_2	400.0	0.253035	0.287694	0.000	0.01850	0.1420	0.39525	1.000
_Chromagram_Mean_3	400.0	0.365098	0.324570	0.000	0.07975	0.2885	0.57650	1.000
_Chromagram_Mean_4	400.0	0.208295	0.253623	0.000	0.01700	0.1050	0.31500	1.000
_Chromagram_Mean_5	400.0	0.350412	0.303521	0.000	0.08975	0.2710	0.53575	1.000
_Chromagram_Mean_6	400.0	0.263880	0.292692	0.000	0.01975	0.1440	0.45050	1.000
_Chromagram_Mean_7	400.0	0.242797	0.275796	0.000	0.02600	0.1410	0.36500	1.000
_Chromagram_Mean_8	400.0	0.391873	0.330826	0.000	0.10200	0.2955	0.63550	1.000
_Chromagram_Mean_9	400.0	0.354632	0.334976	0.000	0.06675	0.2470	0.61200	1.000
_Chromagram_Mean_10	400.0	0.590975	0.357981	0.000	0.26450	0.6120	1.00000	1.000
_Chromagram_Mean_11	400.0	0.342340	0.315808	0.000	0.05950	0.2470	0.56525	1.000
_Chromagram_Mean_12	400.0	0.385620	0.348117	0.000	0.06075	0.2965	0.67075	1.000
_HarmonicChangeDetectionFunction_Mean	400.0	0.328213	0.055520	0.112	0.29075	0.3330	0.36725	0.488
_HarmonicChangeDetectionFunction_Std	400.0	0.192997	0.047092	0.060	0.16000	0.1900	0.22600	0.340
_HarmonicChangeDetectionFunction_Slope	400.0	-0.000157	0.104743	-0.285	-0.05800	-0.0020	0.06325	0.442
_HarmonicChangeDetectionFunction_PeriodFreq	400.0	1.762288	0.930352	0.187	0.96100	1.6820	2.24300	4.486
_HarmonicChangeDetectionFunction_PeriodAmp	400.0	0.769690	0.072107	0.530	0.72500	0.7860	0.82400	0.908
_HarmonicChangeDetectionFunction_PeriodEntropy	400.0	0.966712	0.003841	0.939	0.96500	0.9670	0.96900	0.977

```
[23] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state = 20)
```

```
# Creating model pipeline
pipe = Pipeline([("scaler", preprocessing.StandardScaler()),
                 ("Classifier", SVC(random_state = 20))])
# Searching parameters
params = [{"Classifier__kernel": ["rbf"],
           "Classifier__gamma": [1, 0.1, 0.01, 0.001],
           "Classifier__C": [0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]}
         ]

# grid Search CV
grid_search = GridSearchCV(pipe,
                           params,
                           cv = StratifiedKFold(n_splits = 10,
                                                shuffle = True,
                                                random_state = 20),
                           refit = True,
                           verbose = 2,
                           scoring = "accuracy")

start_time_train = time.time()
grid_search.fit(X_train, y_train)
end_time_train = time.time()

# get the best model
best_model = grid_search.best_estimator_
```

```
# Best parameters
best = pd.DataFrame.from_dict(grid_search.best_params_, orient = "index").rename(columns = {0: "Best"})
best
```

	Best
Classifier__C	1
Classifier__gamma	0.01
Classifier__kernel	rbf

```
[26] y_pred = best_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Doğruluk: {accuracy}")
print("Sınıflandırma Raporu:\n", report)
print(f"eğitim süresi: {(end_time_train - start_time_train) / 60}")
```

```
Doğruluk: 0.8
Sınıflandırma Raporu:
              precision    recall  f1-score   support

       angry       0.95      0.76      0.84        25
       happy       0.81      0.94      0.87        18
       relax       0.82      0.82      0.82        17
         sad       0.64      0.70      0.67        20

    accuracy                           0.80        80
   macro avg       0.80      0.81      0.80        80
weighted avg       0.81      0.80      0.80        80

eğitim süresi: 0.09886568387349447
```

```
# Confusion matrix
svc_cm = confusion_matrix(y_test, y_pred, labels = best_model.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix = svc_cm,
                              display_labels = best_model.classes_)
print("SVC Confusion Matrix")
disp.plot()
plt.show();
```

SVC Confusion Matrix