

dataset_loader

January 11, 2022

available_columns = 'emotion', 'gender', 'subset', 'file_path'

```
[1]: import os
import numpy as np
import pandas as pd

class Loader:
    identifier = None

    @classmethod # returns dataframe with dataset info
    def load_dataset(cls): return None
```

1 Load CREMA-D Normal

```
[2]: class NormalCrema(Loader):
    identifier = 'crema_normal'

    @classmethod
    def load_dataset(cls):
        components=[]
        path = '/data/emo/notebooks/source/datasets/crema'
        for file in os.listdir(path):
            component = np.array(file.replace('.', '_').split('_'))
            component = np.array([component[2], None, None, os.path.join(path,
→file)])
            components.append(component)
        return pd.DataFrame(data=components,
→columns=['emotion', 'gender', 'subset', 'file_path'])
```

2 Load CREMA-D Splitted

```
[3]: class SplittedCrema(Loader):
    identifier = 'cremaSplitted'

    @classmethod
    def load_dataset(cls):
        components=[]
        path = '/data/emo/notebooks/source/datasets/cremaSplitted'
        subsets = ['Test', 'Train', 'Validate']
        for subset in subsets:
            subset_path = f'{path}/{subset}'
            for file in os.listdir(subset_path):
                component = np.array(file.replace('.', '_').split('_'))
                component = np.array([component[2], None, subset, os.path.
→join(subset_path, file)])
                components.append(component)
        return pd.DataFrame(data=components,
→columns=['emotion', 'gender', 'subset', 'file_path'])
```

3 Load RAVDESS Normal

```
[4]: class NormalRavdess(Loader):
    identifier = 'ravdessNormal'
    _emotion_labels = {
        '01': 'neutral',
        '02': 'calm',
        '03': 'happy',
        '04': 'sad',
        '05': 'angry',
        '06': 'fearful',
        '07': 'disgust',
        '08': 'surprised'
    }

    @classmethod
    def load_dataset(cls):
        components=[]
        path = '/data/emo/notebooks/source/datasets/ravdess'
        for file in os.listdir(path):
            component = np.array(file.replace('.', '-').split('-'))
            component = np.array([cls._emotion_labels[component[2]], None,
→None, os.path.join(path, file)])
            components.append(component)
```

```

        return pd.DataFrame(data=components,
        ↪columns=['emotion', 'gender', 'subset', 'file_path'])

```

4 Load RAVDESS Splitted

```

[5]: class SplittedRavdess(Loader):
    identifier = 'ravdessSplitted'
    _emotion_labels = {
        '01': 'neutral',
        '02': 'calm',
        '03': 'happy',
        '04': 'sad',
        '05': 'angry',
        '06': 'fearful',
        '07': 'disgust',
        '08': 'surprised'
    }

    @classmethod
    def load_dataset(cls):
        components=[]
        path = '/data/emo/notebooks/source/datasets/ravdessSplitted'
        subsets = ['Test', 'Train', 'Validate']
        for subset in subsets:
            subset_path = f'{path}/{subset}'
            for file in os.listdir(subset_path):
                component = np.array(file.replace('.', '-').split('-'))
                component = np.array([cls._emotion_labels[component[2]], None,
                ↪subset, os.path.join(subset_path, file)])
                components.append(component)
        return pd.DataFrame(data=components,
        ↪columns=['emotion', 'gender', 'subset', 'file_path'])

```

5 Load CREMA-D Binair

```

[6]: class BinairCrema(Loader):
    identifier = 'crema_binair'

    @classmethod
    def load_dataset(cls):
        components=[]
        path = '/data/emo/notebooks/source/datasets/crema'
        for file in os.listdir(path):

```

```

        component = np.array(file.replace('.', '_').split('_'))
        polarity = 'negative' if component[2] in ['ANG', 'SAD', 'FEA',
↪ 'DIS'] else 'positive'
        component = np.array([polarity, None, None, os.path.join(path,
↪ file)])
        components.append(component)
    return pd.DataFrame(data=components,
↪ columns=['emotion', 'gender', 'subset', 'file_path'])

```

6 Load Male CREMA-D Binair

```

[7]: class MaleSplitCremaBinair(Loader):
        identifier = 'crema_male'
        _crema_d_female_samples =
↪ [1002,1003,1004,1006,1007,1008,1009,1010,1012,1013,1018,1020,1021,1024,1025,1028,1029,1030,
        ↪
↪ 1052,1053,1054,1055,1056,1058,1060,1061,1063,1072,1073,1074,1075,1076,1078,1079,1082,1084,1

        @classmethod
        def load_dataset(cls):
            components=[]
            path = '/data/emo/notebooks/source/datasets/crema'
            for file in os.listdir(path):
                component = np.array(file.replace('.', '_').split('_'))

                if int(component[0]) in cls._crema_d_female_samples:
                    continue

                if component[2] not in ['ANG', 'SAD', 'HAP', 'NEU']:
                    continue

                polarity = ""
                if component[2] in ['ANG', 'SAD']:
                    polarity = 'negative'
                elif component[2] in ['HAP']:
                    polarity = 'positive'
                elif component[2] in ['NEU']:
                    polarity = 'neutral'

                component = np.array([polarity, "Male", None, os.path.join(path,
↪ file)])
                components.append(component)

            return pd.DataFrame(data=components,
↪ columns=['emotion', 'gender', 'subset', 'file_path'])

```

7 Load Female CREMA-D Binair

```
[8]: class FemaleSplitCremaBinair(Loader):
    identifier = 'crema_male'
    _crema_d_female_samples = [
        1002, 1003, 1004, 1006, 1007, 1008, 1009, 1010, 1012, 1013, 1018, 1020, 1021, 1024, 1025, 1028, 1029, 1030,
        1052, 1053, 1054, 1055, 1056, 1058, 1060, 1061, 1063, 1072, 1073, 1074, 1075, 1076, 1078, 1079, 1082, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 17
```

8 Load RAVDESS Binair

TODO: Split Male and Female

```
[9]: class BinairRavdess(Loader):
        identifier = 'ravdess_binair'

        @classmethod
```

```

def load_dataset(cls):
    components=[]
    path = '/data/emo/notebooks/source/datasets/ravdess'
    for file in os.listdir(path):
        component = np.array(file.replace('.', '_').split('_'))
        polarity = 'negative' if component[2] in ['ANG', 'SAD', 'FEA',
↪ 'DIS'] else 'positive'
        component = np.array([polarity, None, None, os.path.join(path,
↪ file)])
        components.append(component)
    return pd.DataFrame(data=components,
↪ columns=['emotion', 'gender', 'subset', 'file_path'])

```

9 Load Male RAVDESS Binair

```

[10]: class MaleBinairRavdess(Loader):
    identifier = 'ravdess_postive_negative_male'
    _emotion_labels = {
        '01': "neutral",
        '02': 'calm',
        '03': 'happy',
        '04': 'sad',
        '05': 'angry',
        '06': 'fearful',
        '07': 'disgust',
        '08': 'surprised'
    }

    @classmethod
    def load_dataset(cls):
        components=[]
        path = '/data/emo/notebooks/source/datasets/ravdess'

        for file in os.listdir(path):
            component = np.array(file.replace('.', '-').split('-'))

            if int(component[6]) % 2 == 0:
                continue

            if component[2] not in ['01', '02', '03', '04', '05']:
                continue

            polarity = ""
            if component[2] in ['05', '04']:
                polarity = 'negative'

```

```

        elif component[2] in ['03', '02']:
            polarity = 'positive'
        elif component[2] in ['01']:
            polarity = 'neutral'

        component = np.array([polarity, "Male", None, os.path.join(path,
↪file)])

        components.append(component)

    return pd.DataFrame(data=components,
↪columns=['emotion', 'gender', 'subset', 'file_path'])

```

10 Load Female RAVDESS Binair

```

[11]: class FemaleBinairRavdess(Loader):
    identifier = 'ravdess_postive_negative_female'
    _emotion_labels = {
        '01': "neutral",
        '02': 'calm',
        '03': 'happy',
        '04': 'sad',
        '05': 'angry',
        '06': 'fearful',
        '07': 'disgust',
        '08': 'surprised'
    }

    @classmethod
    def load_dataset(cls):
        components=[]
        path = '/data/emo/notebooks/source/datasets/ravdess'

        for file in os.listdir(path):
            component = np.array(file.replace('.', '-').split('-'))

            if int(component[6]) % 2 == 0:
                continue

            if component[2] not in ['01', '02', '03', '04', '05']:
                continue

            polarity = ""
            if component[2] in ['05', '04']:
                polarity = 'negative'
            elif component[2] in ['03', '02']:

```

```

        polarity = 'positive'
    elif component[2] in ['01']:
        polarity = 'neutral'

    component = np.array([polarity, "Female", None, os.path.join(path,
↪file)])

    components.append(component)

    return pd.DataFrame(data=components,
↪columns=['emotion', 'gender', 'subset', 'file_path'])

```

11 Load SAVEE Normal

```

[12]: class NormalSavee(Loader):
        identifier = 'savee_normal'

        @classmethod
        def load_dataset(cls):
            emotion = []
            gender = []
            paths = []
            subset = []
            path = '/data/emo/notebooks/source/datasets/savee/'
            for i in os.listdir(path):
                if i[-8:-6] == '_a': emotion.append('angry')
                elif i[-8:-6] == '_d': emotion.append('disgust')
                elif i[-8:-6] == '_f': emotion.append('fear')
                elif i[-8:-6] == '_h': emotion.append('happy')
                elif i[-8:-6] == '_n': emotion.append('neutral')
                elif i[-8:-6] == '_sa': emotion.append('sad')
                elif i[-8:-6] == '_su': emotion.append('surprise')
                paths.append(path + i)
                gender.append('male')
                subset.append(None)
            SAVEE_df = pd.DataFrame(emotion, columns=['emotion'])
            SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(gender,
↪columns=['gender'])], axis=1)
            SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(subset,
↪columns=['subset'])], axis=1)
            SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(paths,
↪columns=['file_path'])], axis=1)

            return pd.DataFrame(data=SAVEE_df,
↪columns=['emotion', 'gender', 'subset', 'file_path'])

```


12 Load SAVEE Splitted

```
[13]: class SplittedSavee(Loader):
    identifier = 'saveeSplitted'

    @classmethod
    def load_dataset(cls):
        emotion = []
        gender = []
        paths = []
        subsetx = []
        path = '/data/emo/notebooks/source/datasets/saveeSplitted'
        subsets = ['Test', 'Train', 'Validate']
        for subset in subsets:
            subset_path = f'{path}/{subset}'
            for i in os.listdir(subset_path):
                if i[-8:-6] == '_a':
                    emotion.append('angry')
                elif i[-8:-6] == '_d':
                    emotion.append('disgust')
                elif i[-8:-6] == '_f':
                    emotion.append('fear')
                elif i[-8:-6] == '_h':
                    emotion.append('happy')
                elif i[-8:-6] == '_n':
                    emotion.append('neutral')
                elif i[-8:-6] == '_sa':
                    emotion.append('sad')
                elif i[-8:-6] == '_su':
                    emotion.append('surprise')
                else:
                    emotion.append('error')
            paths.append(f'{subset_path}/{i}')
            gender.append('male')
            subsetx.append(subset)

        SAVEE_df = pd.DataFrame(emotion, columns=['emotion'])
        SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(gender,
        ↪columns=['gender'])], axis=1)
        SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(subsetx,
        ↪columns=['subset'])], axis=1)
        SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(paths,
        ↪columns=['file_path'])], axis=1)
        return pd.DataFrame(data=SAVEE_df,
        ↪columns=['emotion', 'gender', 'subset', 'file_path'])
```

13 Load TESS Normal

```
[14]: class NormalTess(Loader):
    identifier = 'tess_normal'

    @classmethod
    def load_dataset(cls):
        emotion = []
        gender = []
        paths = []
        subset = []
        path = '/data/emo/notebooks/source/datasets/tess/'
        for i in os.listdir(path):
            fname = os.listdir(path + i)
            for f in fname:
                if i == 'OAF_angry' or i == 'YAF_angry':
                    emotion.append('angry')
                elif i == 'OAF_disgust' or i == 'YAF_disgust':
                    emotion.append('disgust')
                elif i == 'OAF_Fear' or i == 'YAF_fear':
                    emotion.append('fear')
                elif i == 'OAF_happy' or i == 'YAF_happy':
                    emotion.append('happy')
                elif i == 'OAF_neutral' or i == 'YAF_neutral':
                    emotion.append('neutral')
                elif i == 'OAF_Pleasant_surprise' or i == 'YAF_pleasant_surprised':
                    emotion.append('surprise')
                elif i == 'OAF_Sad' or i == 'YAF_sad':
                    emotion.append('sad')
                else:
                    emotion.append('Unknown')
                paths.append(path + i + "/" + f)
                gender.append('female')
                subset.append(None)
            TESS_df = pd.DataFrame(emotion, columns=['emotion'])
            TESS_df = pd.concat([TESS_df, pd.DataFrame(gender,
                columns=['gender'])], axis=1)
            TESS_df = pd.concat([TESS_df, pd.DataFrame(subset,
                columns=['subset'])], axis=1)
            TESS_df = pd.concat([TESS_df, pd.DataFrame(paths,
                columns=['file_path'])], axis=1)

        return pd.DataFrame(data=TESS_df,
            columns=['emotion', 'gender', 'subset', 'file_path'])
```

14 Load TESS Splitted

```
[15]: class SplittedTess(Loader):
    identifier = 'saveeSplitted'

    @classmethod
    def load_dataset(cls):
        emotion = []
        gender = []
        paths = []
        subsetx = []
        path = '/data/emo/notebooks/source/datasets/tessSplitted'
        subsets = ['Test', 'Train', 'Validate']
        for subset in subsets:
            subset_path = f'{path}/{subset}'
            for i in os.listdir(subset_path):
                if 'angry' in i:
                    emotion.append('angry')
                elif 'disgust' in i:
                    emotion.append('disgust')
                elif 'fear' in i:
                    emotion.append('fear')
                elif 'happy' in i:
                    emotion.append('happy')
                elif 'neutral' in i:
                    emotion.append('neutral')
                elif 'surprised' in i:
                    emotion.append('surprise')
                elif 'sad' in i:
                    emotion.append('sad')
                else:
                    emotion.append('Unknown')
            paths.append(f'{subset_path}/{i}')
            gender.append('female')
            subsetx.append(subset)
        TESS_df = pd.DataFrame(emotion, columns=['emotion'])
        TESS_df = pd.concat([TESS_df, pd.DataFrame(gender,
        ↪columns=['gender'])], axis=1)
        TESS_df = pd.concat([TESS_df, pd.DataFrame(subsetx,
        ↪columns=['subset'])], axis=1)
        TESS_df = pd.concat([TESS_df, pd.DataFrame(paths,
        ↪columns=['file_path'])], axis=1)
        return pd.DataFrame(data=TESS_df,
        ↪columns=['emotion', 'gender', 'subset', 'file_path'])
```

15 Load Quaternair Combined

```
[16]: class QuaternairCombined(Loader):
        identifier = 'combined_quaternair'

        @classmethod
        def load_dataset(cls):
            loaded_dataset_1 = NormalCrema.load_dataset()
            loaded_dataset_2 = NormalRavdess.load_dataset()
            loaded_dataset_3 = NormalSavee.load_dataset()
            loaded_dataset_4 = NormalTess.load_dataset()
            dataset = pd.concat([loaded_dataset_1, loaded_dataset_2,
                                loaded_dataset_3, loaded_dataset_4],
                                ignore_index=True, sort=False)
            for index, value in dataset.iterrows():
                if value['emotion'] in ['SAD', 'sad']: value['emotion'] = 'sad'
                if value['emotion'] in ['ANG', 'angry']: value['emotion'] = 'angry'
                if value['emotion'] in ['NEU', 'neutral']: value['emotion'] =
                ↪ 'neutral'
                if value['emotion'] in ['HAP', 'happy']: value['emotion'] = 'happy'
            return dataset.loc[dataset['emotion'].isin(['angry', 'neutral',
                ↪ 'happy', 'sad'])]
```

16 Load Trinair Combined

```
[17]: class TrinairCombinedPN(Loader):
        identifier = 'combined_trinair'

        @classmethod
        def load_dataset(cls):
            loaded_dataset_1 = NormalCrema.load_dataset()
            loaded_dataset_2 = NormalRavdess.load_dataset()
            loaded_dataset_3 = NormalSavee.load_dataset()
            loaded_dataset_4 = NormalTess.load_dataset()
            dataset = pd.concat([loaded_dataset_1, loaded_dataset_2,
                                loaded_dataset_3, loaded_dataset_4],
                                ignore_index=True, sort=False)
            for index, value in dataset.iterrows():
                if value['emotion'] in ['ANG', 'angry', 'SAD', 'sad']:
                    value['emotion'] = 'negative'
                elif value['emotion'] in ['HAP', 'happy', 'CAL', 'calm']:
                    value['emotion'] = 'positive'
                elif value['emotion'] in ['NEU', 'neutral']:
                    value['emotion'] = 'neutral'
```

```

        return dataset.loc[dataset['emotion'].isin(['negative', 'positive',
↪ 'neutral'])]

```

17 Load Quaternair Combined Splitted

```

[18]: class QuaternairCombinedSplitted(Loader):
        identifier = 'combined_quaternairSplitted'

        @classmethod
        def load_dataset(cls):
            loaded_dataset_1 = SplittedCrema.load_dataset()
            loaded_dataset_2 = SplittedRavdess.load_dataset()
            loaded_dataset_3 = SplittedSavee.load_dataset()
            loaded_dataset_4 = SplittedTess.load_dataset()
            dataset = pd.concat([loaded_dataset_1, loaded_dataset_2,
↪                               loaded_dataset_3, loaded_dataset_4],
                                ignore_index=True, sort=False)
            for index, value in dataset.iterrows():
                if value['emotion'] in ['SAD', 'sad']: value['emotion'] = 'sad'
                if value['emotion'] in ['ANG', 'angry']: value['emotion'] = 'angry'
                if value['emotion'] in ['NEU', 'neutral']: value['emotion'] =
↪ 'neutral'
                if value['emotion'] in ['HAP', 'happy']: value['emotion'] = 'happy'
            return dataset.loc[dataset['emotion'].isin(['angry', 'neutral',
↪ 'happy', 'sad'])]

```

18 Load Male Quaternair Combined

```

[19]: class QuaternairMaleCombinedPN(Loader):
        identifier = 'combined_quaternair_Male_Positive_Negative'

        @classmethod
        def load_dataset(cls):
            components = []
            loaded_dataset_1 = NormalCrema.load_dataset()
            loaded_dataset_2 = NormalRavdess.load_dataset()
            loaded_dataset_3 = NormalSavee.load_dataset()
            loaded_dataset_4 = NormalTess.load_dataset()
            dataset = pd.concat([loaded_dataset_1, loaded_dataset_2,
↪                               loaded_dataset_3, loaded_dataset_4],
                                ignore_index=True, sort=False)

```

```

for index, value in dataset.iterrows():
    if value['emotion'] in ['ANG', 'angry', 'SAD', 'sad']:
        value['emotion'] = 'negative'
    elif value['emotion'] in ['HAP', 'happy', 'CAL', 'calm']:
        value['emotion'] = 'positive'
    elif value['emotion'] in ['NEU', 'neutral']:
        value['emotion'] = 'neutral'

components = np.array([value, "Male", None, os.path.join(path, file)])
components.append(component)

return dataset.loc[dataset['emotion'].isin(['negative', 'positive',
↪ 'neutral'])],

    #return pd.DataFrame(data=components,
↪ columns=['emotion', 'gender', 'subset', 'file_path'])

```

19 Load Female Quaternair Combined

```

[20]: class QuaternairFemaleCombinedPN(Loader):
    identifier = 'combined_quaternair_Female_Positive_Negative'

    @classmethod
    def load_dataset(cls):
        components = []
        loaded_dataset_1 = NormalCrema.load_dataset()
        loaded_dataset_2 = NormalRavdess.load_dataset()
        loaded_dataset_3 = NormalSavee.load_dataset()
        loaded_dataset_4 = NormalTess.load_dataset()
        dataset = pd.concat([loaded_dataset_1, loaded_dataset_2,
                               loaded_dataset_3, loaded_dataset_4],
↪ ignore_index=True, sort=False)

        for index, value in dataset.iterrows():
            if value['emotion'] in ['ANG', 'angry', 'SAD', 'sad']:
                value['emotion'] = 'negative'
            elif value['emotion'] in ['HAP', 'happy', 'CAL', 'calm']:
                value['emotion'] = 'positive'
            elif value['emotion'] in ['NEU', 'neutral']:
                value['emotion'] = 'neutral'

            components = np.array([value, "Female", None, os.path.join(path,
↪ file)])
            components.append(component)

```

```

        return dataset.loc[dataset['emotion'].isin(['negative', 'positive',
↪ 'neutral'])],
        #return pd.DataFrame(data=components,
↪ columns=['emotion', 'gender', 'subset', 'file_path'])

```

20 Load Single Record

```

[21]: class SingleValue(Loader):
        identifier = 'single_value'

        @classmethod
        def load_dataset(cls):
            components=[]
            path = '/data/emo/notebooks/source/datasets/crema'
            for file in os.listdir(path):
                component = np.array(file.replace('.', '_').split('_'))
#                component = np.array([component[2], None, None, os.path.
↪ join(path, file)])
                component = np.array(["Unknown", None, None, os.path.join(path,
↪ file)])
                components.append(component)
                break

            if component[0] in ['ANG', 'HAP', 'SAD', 'NEU']:
                components.append(component)
                break

            print(components)
            return pd.DataFrame(data=components,
↪ columns=['emotion', 'gender', 'subset', 'file_path'])

```

	emotion	gender	subset	\
0	happy	None	Test	
1	fearful	None	Test	
2	fearful	None	Test	
3	angry	None	Test	
4	happy	None	Test	
...	
3763	fearful	None	Validate	
3764	angry	None	Validate	
3765	disgust	None	Validate	
3766	happy	None	Validate	

3767 angry None Validate

```

                                     file_path
0      /data/emo/notebooks/source/datasets/ravdess_sp...
1      /data/emo/notebooks/source/datasets/ravdess_sp...
2      /data/emo/notebooks/source/datasets/ravdess_sp...
3      /data/emo/notebooks/source/datasets/ravdess_sp...
4      /data/emo/notebooks/source/datasets/ravdess_sp...
...
3763   /data/emo/notebooks/source/datasets/ravdess_sp...
3764   /data/emo/notebooks/source/datasets/ravdess_sp...
3765   /data/emo/notebooks/source/datasets/ravdess_sp...
3766   /data/emo/notebooks/source/datasets/ravdess_sp...
3767   /data/emo/notebooks/source/datasets/ravdess_sp...
```

[3768 rows x 4 columns]