Documentatie Proiect Machine Learning

Pentru inceput, trebuie sa discutam despre librarii. Am folosit librosa pentru a incarca fisierele audio de tip .wav si pentru a extrage features, libraria os pentru path-urile diverselor fisiere, matplotlib pentru plotarea graficelor, numpy pentru stocare si reshape-uri, pandas pentru functionalitatea Data Frame-urilor si a Series-urilor si a functiilor apply(), to_pickle(), read_pickle(), din sklearn am importat diversele modele, cat si preprocesarea si diversele functii de metrics, iar din Keras am folosit modelul Sequential, layerele de tip Danse, Dropout si Activation si Utils.

```
def normalize(train_features, test_features):
    scaler = preprocessing.StandardScaler()
    scaler.fit(train_features)
    scaled_train_feats = scaler.transform(train_features)
    scaled_test_feats = scaler.transform(test_features)

    return scaled_train_feats, scaled_test_feats
```

Prima functie folosita este functia Normalize, unde folosind StandardScaler() ca sa standardizam datele(procesul prin care punem diferite variabile pe aceasi scara. Trebuie scazut din fiecare variabila media si impartita la deviata standard a sa. Acest proces se intampla cand se apeleaza functia fit, iar pentru scaler.trasnform se centreaza si scaleaza datele.)

```
def get_features_from_train(row):
    file_name = os.path.join(os.path.abspath(path_train), str(row[0]))
    audio_train, sfreq_train = lb.load(file_name, res_type='kaiser_fast')
    "Le si scalam"
    mfccs = np.mean(lb.feature.mfcc(y=audio_train, sr=sfreq_train, n_mfcc=40).T, axis=0)

    labels = row[1]
    return pd.Series([mfccs, labels])
```

Urmeaza functia de extragere a features. Aceasta (cu diverse modificari pentru fisierele de test, train si validation) o apelez de 3 ori. Extragem pentru fiecare fisier, cate un nume de audio file. Folosim functia de load din librosa pentru a incarca fisierul audio ca un floating point time series. Am extras Mel-Frequency Cpestral Coefficients din time series si ii calculam media temporala.
Am ales MFCC deoarece este folosita in principal pentru speech recognition si am testat prin trial and error(am folosit majoritatea functiilor si mfcc dadea cel mai bine).

```
train_data = train.apply(get_features_from_train, axis=1)
train_data.columns = ['feature', 'label']
train_data.to_pickle("./train.pkl")
```

Folosind apply(nume_functie, axis=1) aplicam functia respectiva pe fiecare rand al fisierului, ii adaugam coloanele pentru features si label(pentru a le putea extrage usor mai tarziu) si rezultatul il transformam intr-un fisier de tip pickle pentru a nu repeta procesul de fiecare date cand rulam codul. Dupa ce am rulat o data il comentez,

```
train_data = pd.read_pickle("./train.pkl")
x = np.array(train_data.feature.tolist())
y = np.array(train_data.label.tolist())
y_binary = to_categorical(y)
```

Citim cu read_pickle din fisierul creat mai sus, transformam intr un nparray, iar cu functia to_categorical din keras transformam labels intr o matrice binara(1 si 0 unde este nevoie)

```
"Initial folosim un model Sequential pentru ca dorim sa facem un stack de layere cu un singur input si un singur output"
"MLP ul meu are 3 hidden layere"
model = Sequential()

"Primul layer are 512 output neurons"
"Deoarece x_train are shape ul (8000, 40) vom folosi ca input (40, )"
"Se activeaza folosind Rectified Linear Unit si are un dropout de 0.4 pentru a reduce overfitting-ul"
model.add(Dense(512, input_shape=(40, )))
model.add(Activation('relu'))
model.add(Dropout(0.4))

"Layer ul 2 are tot 512 output neurons, aceasi activare si acelasi dropout"
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.4))

"Layerul final, cel de output are 2 output neurons deoarece avem 2 label uri in problema noastra."
"Am folosit activarea softmax fara dropout"
model.add(Dense(2))
model.add(Activation('softmax'))
```

Dupa cum scrie si in comentarii am creat un MLP folosind modelul Sequential pentru a da stack la 3 layere. Primele doua layere au cate 512 neurons. Am folosit Droput de 0.4 pentru a incerca pe cat posibil sa reduc overfitting-ul, dar si a mentine o acuratete mai mare. Am folosit pentru ambele functia de activare ReLU(am incercat si Leaky ReLU, dar nu am vazut o schimbare mare). Ultimul layer are ca activation function Softmax(de obicei Softmax se foloseste pentru layer-ul de output) si avand doar 2 neuroni de output deoarece predictiile noastre trebuiau sa fie 0 sau 1. Am folosit keras si nu mlp-ul din sklearn pentru o acuratete mai mare.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               20992
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 1026
_________________________________________________________________
activation_3 (Activation)    (None, 2)                 0
=================================================================
```

Acesta este model.summary()

```
model.compile(loss='binary_crossentropy', metrics=['accuracy'], optimizer='adam')

num_epochs = 50
num_batch_size = 64

history_mlp = model.fit(x_std, y_binary, batch_size=num_batch_size, validation_data=(x_val_std, y_val_binary), epochs=num_epochs, verbose=1)

"Acum incepe plotarea pentru acuratea si loss a mlp ului"
```

Configuram modelul si incepem sa antrenam. Am folosit functia de loss binary_crossentropy deoarece avem de distins intre 2 clase, metrics = accuracy pentru ca ne intereseaza acuratetea, iar optimizer = Adam deoarece am avut cea mai mare acuratete cu el. Am incercat sa il mai si modific(learning_rate si beta1, beta2), sau sa folosesc rmsprop dar nu am avut foarte mare succes. Am incercat cu mai multe batch_size uri dar am avut acuratete buna cu 64, iar numarul de epoci 50 este deja mare deoarece are deja overfit.

Am folosit plot pentru a crea graficele pentru loss, val_loss, acc si val_acc folosind istoricul modelului, iar folosind model.predict_classes generam predictiile(doar 0 si 1). Cu un pandas DataFrame le scriem intr-un CSV, avand headers.

```
accuracy = accuracy_score(y_val, y_pred)
f1 = f1_score(y_val, y_pred)
recall = recall_score(y_val, y_pred)
precision = precision_score(y_val, y_pred)
c_matrx = confusion_matrix(y_val, y_pred)

print(f"accuracy: {accuracy}, f1_score: {f1}, recall: {recall} si precision: {precision}")
print("Matricea de confuzie:")
print(c_matrx)

"Scoatem predictiile si le scriem in csv cu ajutorul pandas"
y_pred = model.predict_classes(x_tests)

df = pd.DataFrame({'name': file_names[1:], 'label': y_pred})
df.to_csv('predictions.csv', index=False)
```

Cu ajutorul functiilor de scor generam acuratetea, f1_score, recall, precision si confusion matrix.

Pentru o comparatie am folosit un SVM cu kernel linear si C=10. Am folosit aceleasi functii de scor ca si la MLP. Rezultatul este in favoarea MLP ului.

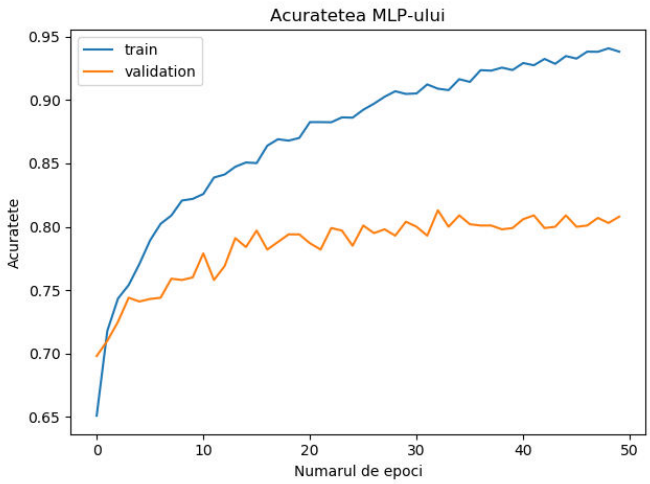
```
svm = SVC(kernel="linear", C=10)

svm.fit(x_std, y)

y_pred = svm.predict(x_val_std)
```

MLP:
accuracy: 0.808, f1_score: 0.8218923933209646, recall: 0.8390151515151515 si precision: 0.8054545454545454
Matricea de confuzie:

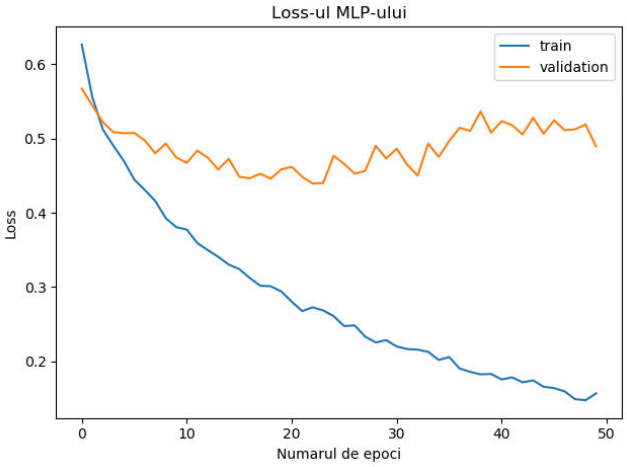
```
mlp = MLPClassifier(max_iter=5000, activation='relu', alpha=0.0005, hidden_layer_sizes=(555, 100, 50), learning_rate='constant', solver='adam')

mlp.fit(x_std, y)

y_pred = mlp.predict(x_val_std)
```

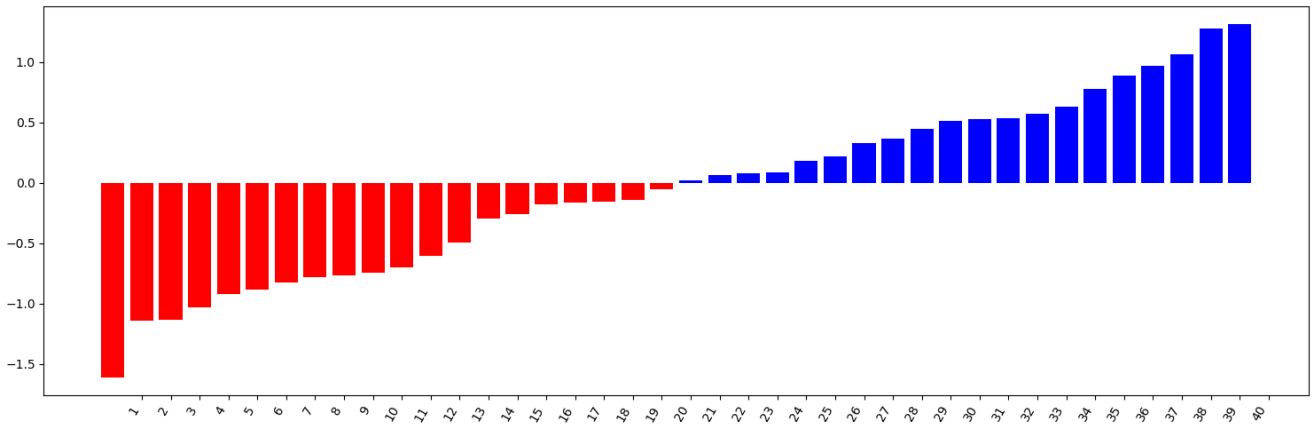
[[365 107]
 [85 443]]

SVM:
accuracy: 0.641, f1_score: 0.6571155682903533, recall: 0.6515151515151515 si precision: 0.6628131021194605



Matricea de confuzie:
[[297 175]
 [184 344]]





Se poate observa clar overfitt-ul.

Plotarea coeficientilor pentru SVM-ul de mai sus.
Celelalte modele pe care le-am folosit sunt:
-Un MLP din sklearn. Am incercat sa fac un GridSearchCV pe el si acestia sunt parametrii pe care i-am gasit.
Am obinut:
accuracy: 0.784, f1_score: 0.7927063339731286, recall: 0.7821969696969697 si precision: 0.8035019455252919
Matricea de confuzie:
[[371 101]
 [115 413]]
-XGBoost. Nu cred ca este foarte bine optimizat, cred ca putea lua o acuratete mai mare. Din ce am observat facea overfit destul de mare si era foarte greu de facut parameter tunning

```
xgb = XGBClassifier(subsample=0.8, n_estimators=2000, min_child_weight=1, max_depth=5,
                    learning_rate=0.1, gamma=0.5, colsample_bytree=1, objective='binary:logistic')

xgb.fit(x_std, y)

y_pred = xgb.predict(x_val_std)
y_pred = [round(value) for value in y_pred]
```

accuracy: 0.764, f1_score: 0.774378585086042, recall: 0.7670454545454546 si precision: 0.7818532818532818
Matricea de confuzie:
[[359 113]
 [123 405]]

-KNN cu numar de neighbors=15

```
knn = KNeighborsClassifier(n_neighbors=15)

knn.fit(x_std, y)

y_pred = knn.predict(x_val_std)
```

accuracy: 0.748, f1_score: 0.7692307692307692, recall: 0.7954545454545454 si precision: 0.7446808510638298
Matricea de confuzie:
[[328 144]
 [108 420]]
-RandomForest care nu a fost foarte bine optimizat

```
clf = RandomForestClassifier(min_samples_split=2, min_samples_leaf=1, n_estimators=200)

clf.fit(x_std, y)

y_pred = clf.predict(x_val_std)
```

accuracy: 0.73, f1_score: 0.737864077669903, recall: 0.7196969696969697 si precision: 0.7569721115537849
Matricea de confuzie:
[[350 122]
 [148 380]]
-Un SVM mai bine optimizat decat cel facut pentru comparatie.

```
svm = SVC(C=10)

svm.fit(x_std, y)

y_pred = svm.predict(x_val_std)
```

accuracy: 0.79, f1_score: 0.7984644913627639, recall: 0.7878787878787878 si precision: 0.8093385214007782
Matricea de confuzie:
[[374 98]
 [112 416]]
-Un CNN care din pacate avea overfit extrem de mare(minus la loss) si nu am mai incercat sa il rezolv.

Dintre toate modele cele mai bune pentru mine au fost MLP(ambele cazuri) si SVM. Din pacate nu am reusit sa optimizez mai bine SVM-ul deoarece dureaza extrem de mult pe laptop-ul meu.