Proyecto Final Statistical Learning 2

Anonymous Author(s)
Affiliation
Address
email

Abstract

A continuación se presenta la información del proyecto final de Statistcal Learning, en el cual se desarrollaron tres problemas MLP, CNN y Análisis de sentimientos.

1 Feed Forward Network

Para feedforward se trabajo con un dataset para la detección de malware en base a sequencias de llamadas API.

```
SM_filepath = './MLP'
ds_Malware0 = pd.read_csv("./dynamic_api_call_sequence_per_malware_100_0_306.csv")
ds_Malware0.head()
```

	hash	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	...	t_91	t_92	t_93	t_94	t_95	t_96	t_97
0	071e8c3f8922e186e57548cd4c703a5d	112	274	158	215	274	158	215	298	76	...	71	297	135	171	215	35	208
1	33f8e6d08a6aae939f25a8e0d63dd523	82	208	187	208	172	117	172	117	172	...	81	240	117	71	297	135	171
2	b68abd064e975e1c6d5f25e748663076	16	110	240	117	240	117	240	117	240	...	65	112	123	65	112	123	65
3	72049be7bd30ea61297ea624ae198067	82	208	187	208	172	117	172	117	172	...	208	302	208	302	187	208	302
4	c9b3700a77facf29172f32df6bc77f48	82	240	117	240	117	240	117	240	117	...	209	260	40	209	260	141	260

5 rows × 102 columns

1.1 Limpieza de Dataset

Antes de procesar la informacion en el modelo, se procedio a eliminar la inforamción que no aporta nada a el modelo.

```
#ELIMINACION DE FEATURES QUE NO APORTAN INFORMACION
ds_Malware = ds_Malware0.drop(columns=["hash"],axis=1)
ds_Malware.shape
```

```
(43876, 101)
```

```
Y= ds_Malware["malware"]
X = ds_Malware.drop(columns=["malware"])
X_Train, X_Test, Y_Train, Y_Test = TTS(X,Y,test_size=0.2,random_state=9)
```

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

1.2 Definición del Modelo

Se utilizaron dos metodos para la generación del modelo Keras y Sklearn con MLP (Multi Layer Perceptron).

- Función de activacion = relu y sigmoid - Optimizador = Adam - Loss = Binary Cross Entropy

UTILIZANDO KERAS

```
]: model = Sequential()
model.add(Dense(128, input_shape=(100,), activation='relu')),
model.add(Dropout(0.5)),
model.add(Dense(64, activation='relu')),
model.add(Dropout(0.5)),
model.add(Dense(32, activation='relu')),
model.add(Dropout(0.5)),
model.add(Dense(16, activation='relu')),
model.add(Dropout(0.5)),
model.add(Dense(8, activation='relu')),
model.add(Dropout(0.5)),
model.add(Dense(4, activation='relu')),
model.add(Dropout(0.5)),
model.add(Dense(1, activation='sigmoid'))
#model.add(Dense(2, activation='softmax'))
```

```
]: Model_Optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(optimizer=Model_Optimizer, loss=tf.keras.losses.BinaryCrossentropy(), metrics=['
Model_EarlyStop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', min_delta=0, patien
Model_checkpoint= tf.keras.callbacks.ModelCheckpoint(filepath=SM_filepath,save_weights_only=Fa
```

1.3 Resultados

Se puede observar que al correr el modelo se obtiene un accuracy del 97

```
[29]: Pred_K_P = model.predict(X_Test)
Accuracy_K =metrics.average_precision_score(Y_Test, Pred_K_P)
MLP_K = pd.DataFrame({
    'MLP': ['MLP Keras'],
    'Acurracy': [Accuracy_K]
})
MLP_K
```

[29]:

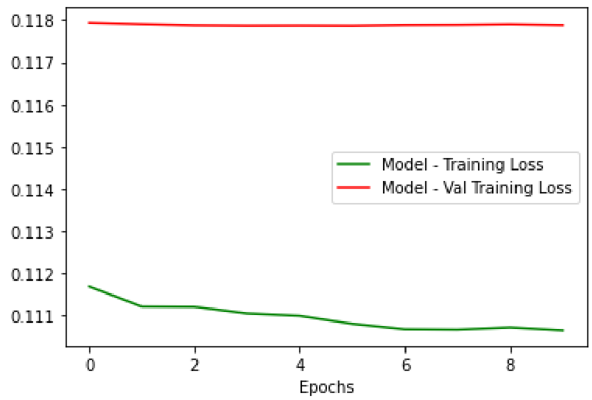
	MLP	Acurracy
0	MLP Keras	0.971399

```python
Model_Fit = model.fit(X_Train.values, Y_Train.values, epochs = 10, batch_size=1000, validation
Model_Fit_History = Model_Fit.history
save_model(model, SM_filepath)
```

```
INFO:tensorflow:Assets written to: ./MLP\assets
```

```python
Loss = Model_Fit_History['loss']
Val_Loss = Model_Fit_History['val_loss']
plt.plot(Loss,'g',label='Model - Training Loss')
plt.plot(Val_Loss,'r',label='Model - Val Training Loss')
plt.legend()
plt.xlabel("Epochs")
```

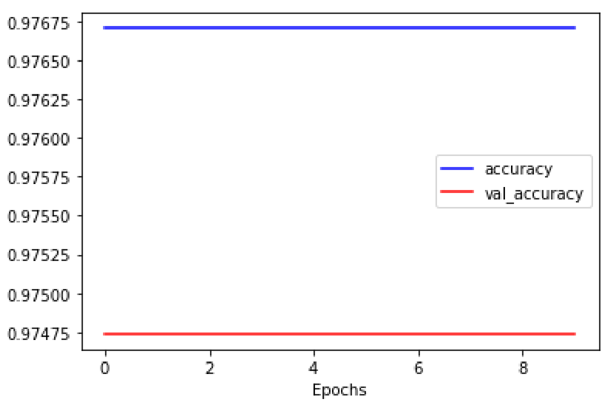
```
Text(0.5, 0, 'Epochs')
```



En base al training realizado

```python
Accuracy = Model_Fit_History['accuracy']
Val_Accuracy = Model_Fit_History['val_accuracy']
plt.plot(Accuracy,'-b',label='accuracy')
plt.plot(Val_Accuracy,'-r',label='val_accuracy')
plt.legend()
plt.xlabel("Epochs")
```

```
Text(0.5, 0, 'Epochs')
```



Ahora utilzando sklearn se obtien de igual manera un accurracy aceptable un poco mayor que con keras

UTILIZANDO SKLEARN

```
MLP = MLPC(solver='adam',hidden_layer_sizes=(100,100,100),max_iter=800,random_state=50)
MLP.fit(X_Train, Y_Train)
Pred = MLP.predict(X_Test)
Pred
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

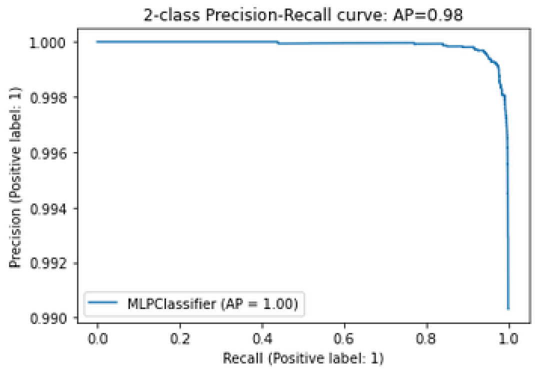
```
F1 = metrics.f1_score(Y_Test, Pred)
Accuracy =metrics.accuracy_score(Y_Test, Pred)
Precission = metrics.precision_score(Y_Test, Pred)
Recall = metrics.recall_score(Y_Test, Pred)
MLP_S = pd.DataFrame({
    'MLP': ['MLP Sklearn'],
    'F1': [F1],
    'Acurracy': [Accuracy],
    'Precission': [Precission],
    'Recall': [Recall]
})
MLP_S
```

	MLP	F1	Acurracy	Precission	Recall
0	MLP Sklearn	0.991214	0.982794	0.983376	0.999179

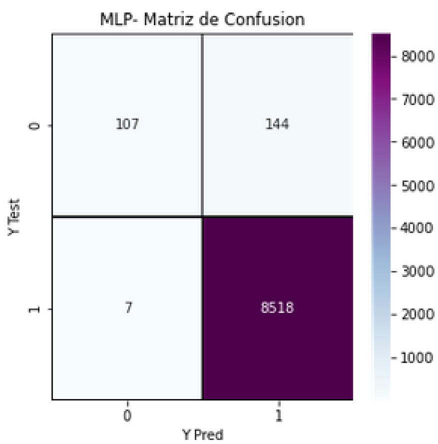
23

```
average_precision = average_precision_score(Y_Test, Pred)
disp = plot_precision_recall_curve(MLP,X_Train, Y_Train)
disp.ax_.set_title('2-class Precision-Recall curve: ''AP={0:0.2f}'.format(average_precision))

print('Average precision-recall score: {0:0.2f}'.format(average_precision))
```

```
Average precision-recall score: 0.98
```




```
ConM = confusion_matrix(Y_Test, Pred)
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(ConM, annot=True, linewidth=0.4, linecolor='black', fmt='g', ax=ax, cmap="BuPu")
plt.title('MLP- Matriz de Confusion')
plt.xlabel('Y Pred')
plt.ylabel('Y Test')
plt.show()
```



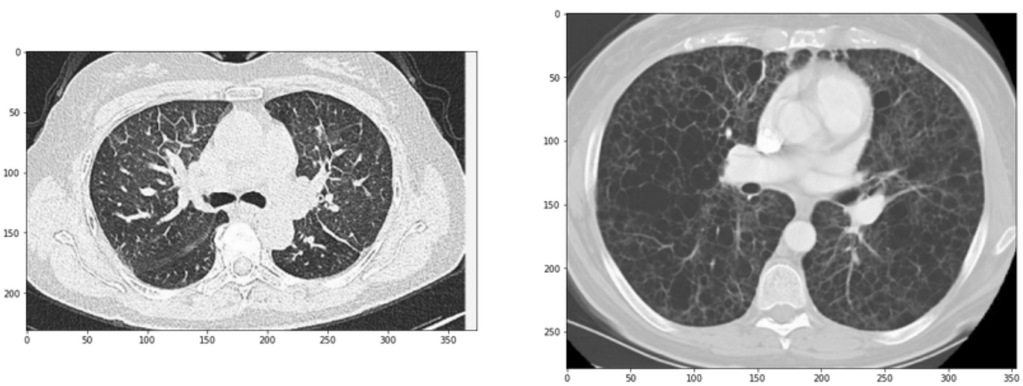
2 Convolutional Network

Para este problema se utilizo un data set para la deteccion de pacientes que tuvieron o no covid por medio de el analisis de imagenes de los pulmones de los pacientes.

```python
SM_filepath = './MLP'
Directorio = './COVID/'
ds_PulmonCovidP = os.path.join('./CT_COVID/')
ds_PulmonCovidN = os.path.join('./CT_NonCOVID/')
print('Total Casos Positivos :{}'.format(len(os.listdir(ds_PulmonCovidP))))
print('Total Casos Negativos:{}'.format(len(os.listdir(ds_PulmonCovidN))))

CovPositivo = glob(os.path.join(ds_PulmonCovidP,"*.png"))
CovNegativo = glob(os.path.join(ds_PulmonCovidN,"*.png"))
CovNegativo.extend(glob(os.path.join(ds_PulmonCovidN,"*.jpg")))
#MUESTA DE IMAGENES
IMG_P = cv2.imread(os.path.join(CovPositivo[150]))
IMG_N = cv2.imread(os.path.join(CovNegativo[150]))
f = plt.figure(figsize=(20, 20))
f.add_subplot(1, 2, 1)
plt.imshow(IMG_P)
f.add_subplot(1, 2, 2)
plt.imshow(IMG_N)
```

```
Total Casos Positivos :349
Total Casos Negativos:397
```

```
<matplotlib.image.AxesImage at 0x1e1810cc370>
```



2.1 Resultados

Se realizaron distintas pruebas para encontrar el modelo el cual presentarar una mejor predicción

KERAS - Test 1

```
Epoch = 40
Batch = 64
Img_H, Img_W = 248,248
opt = tf.keras.optimizers.Adam(lr=0.001,decay=0.001/Epoch)
es = EarlyStopping(monitor='val_acc', mode='max', verbose=1, patience=10, restore_best_weights=True)
```

```
Train_D = ImageDataGenerator(rescale=1./255,horizontal_flip=True,rotation_range=5,width_shift_range=0.05,height_shi
Train_G = Train_D.flow_from_directory(Directorio,target_size=(Img_H, Img_W),batch_size=Batch,class_mode='binary', c
Val_G = Train_D.flow_from_directory(Directorio, target_size=(Img_H, Img_W),batch_size=Batch,class_mode='binary',col
```

```
Found 598 images belonging to 2 classes.
Found 148 images belonging to 2 classes.
```

```
model = Sequential()
model.add(Conv2D(32, 3, padding='same', activation='relu',input_shape=(Img_H, Img_W, 1)))
model.add(MaxPool2D())
model.add(Conv2D(64, 5, padding='same', activation='relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(optimizer=opt, loss=keras.losses.binary_crossentropy, metrics=['accuracy', 'Precision', 'Recall'])
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 248, 248, 32)      320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 124, 124, 32)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 124, 124, 64)      51264
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 62, 62, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 246016)            0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               31490176
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 31,541,889
Trainable params: 31,541,889
Non-trainable params: 0
_________________________________________________________________
```

31

KERAS - Test 2

```
Epoch2 = 50
Batch2 = 70
opt2 = tf.keras.optimizers.Adam(lr=0.001,decay=0.001/Epoch2)
es2 = EarlyStopping(monitor='val_acc', mode='max', verbose=1, patience=10, restore_best_weights=True)
```

```
Train_D = ImageDataGenerator(rescale=1./255,horizontal_flip=True,rotation_range=5,width_shift_range=0.05,height_shift_range=0.05,shear_range=0.05,zoom_range=0.05,va
Train_G = Train_D.flow_from_directory(Directorio,target_size=(Img_H, Img_W),batch_size=Batch,class_mode='binary', color_mode="grayscale",subset='training')
Val_G = Train_D.flow_from_directory(Directorio, target_size=(Img_H, Img_W),batch_size=Batch,class_mode='binary',color_mode="grayscale", subset='validation')
```

```
model2 = Sequential()
model2.add(Conv2D(16, 1, padding='same', activation='relu',input_shape=(Img_H, Img_W, 1)))
model2.add(MaxPool2D())
model2.add(Conv2D(32, 3, padding='same', activation='relu'))
model2.add(MaxPool2D())
model2.add(Conv2D(64, 5, padding='same', activation='relu'))
model2.add(MaxPool2D())
model2.add(Flatten())
model2.add(Dense(128, activation='relu'))
model2.add(Dropout(0.5))
model2.add(Dense(64, activation='relu'))
model2.add(Dropout(0.4))
model2.add(Dense(8, activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(1, activation='sigmoid'))
```

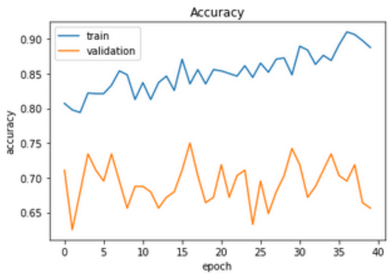
```
model2.compile(optimizer=opt2, loss=keras.losses.binary_crossentropy, metrics=['accuracy', 'Precision', 'Recall'])
```

```
model2.summary()
```

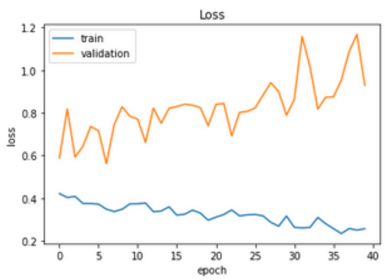
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 248, 248, 16)      32
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 124, 124, 16)      0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 124, 124, 32)      4640
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 62, 62, 64)        51264
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 31, 31, 64)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 61504)             0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               7872640
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_7 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_13 (Dense)             (None, 8)                 520
_________________________________________________________________
dropout_8 (Dropout)          (None, 8)                 0
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 9
```

Danod mejores resutlados el segundo test de keras

```
plt.title('Model - Accuracy')
plt.plot(Model_Fit.history['accuracy'])
plt.plot(Model_Fit.history['val_accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```




```
plt.title('Model - Loss')
plt.plot(Model_Fit.history['loss'])
plt.plot(Model_Fit.history['val_loss'])
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

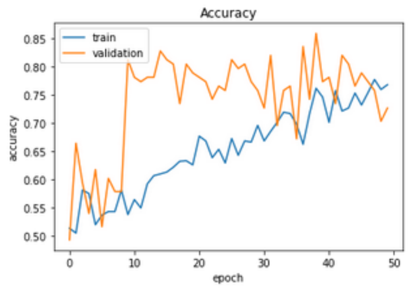



```
y_pred = (model.predict_generator(Val_G) > 0.5).astype(int)
y_true = Val_G.classes

for name, value in zip(model.metrics_names, model.evaluate_generator(Val_G)):
    print(f'{name}: {value}')

print(f'F1 score: {sklearn.metrics.f1_score(y_true, y_pred)}')
```

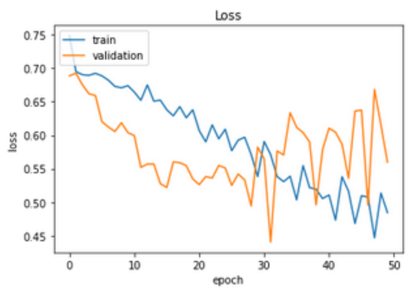
```
loss: 0.9671728610992432
accuracy: 0.662162184715271
precision: 0.6494845151901245
recall: 0.797468364238739
F1 score: 0.576271186440678
```

34 dense_14 (Dense) (None, 1) 9

```
plt.title('Accuracy')
plt.plot(hist2.history['accuracy'])
plt.plot(hist2.history['val_accuracy'])
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```




```
plt.title('Loss')
plt.plot(hist2.history['loss'])
plt.plot(hist2.history['val_loss'])
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```




```
y_pred2 = (model2.predict_generator(Val_G) > 0.5).astype(int)
y_true2 = Val_G.classes

for name, value in zip(model2.metrics_names, model2.evaluate_generator(Val_G)):
    print(f'{name}: {value}')

print(f'F1 score: {sklearn.metrics.f1_score(y_true, y_pred)}')
```

```
loss: 0.6191083192825317
accuracy: 0.7432432174682617
precision: 0.7252747416496277
recall: 0.8354430198669434
F1 score: 0.576271186440678
```

3 Recurrent Neuronal Network

Para la red recurrente se utilizo un data set de Yelp, con comentarios de usuarios en donde 1 es Negativo y 2 representa Positivo.

```
train = pd.read_csv('./train.csv', names = ['sentiment', 'text'] )
test  = pd.read_csv('./test.csv',  names = ['sentiment', 'text'] )
train.head()
```

	sentiment	text
0	1	Unfortunately, the frustration of being Dr. Go...
1	2	Been going to Dr. Goldberg for over 10 years. ...
2	1	I don't know what Dr. Goldberg was like before...
3	1	I'm writing this review to give you a heads up...
4	2	All the food is great here. But the best thing...

Se procedio procesar la informacion y a definir el modelo para predecir.