Recommendation of Refactoring Techniques to address Self-Admitted Technical Debt

CNN Documentation

- recall_m
- precision_m
- f1_m
- CNN

built with pdoc

CNN

def recall_m(y_true, y_pred):

View Source

```
def recall_m(y_true, y_pred):
    """
    Metrics have been removed from Keras core. we need to calculate them manually
    here we will calculate the recall score
    :param y_true:is the true data (or target, ground truth)
    :param y_pred:is the data predicted (calculated, output) by your model.
    :return: recall score
    """
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall
```

Metrics have been removed from Keras core. we need to calculate them manually here we will calculate the recall score :param y_true:is the true data (or target, ground truth) :param y_pred:is the data predicted (calculated, output) by your model. :return: recall score

def precision_m(y_true, y_pred):

View Source

```
def precision_m(y_true, y_pred):
    """
    Metrics have been removed from Keras core. we need to calculate them manually
    here we will calculate the precision score
    :param y_true:is the true data (or target, ground truth)
    :param y_pred:is the data predicted (calculated, output) by your model.
    :return: precision score
    """
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision
```

Metrics have been removed from Keras core. we need to calculate them manually here we will calculate the precision score :param y_true:is the true data (or target, ground truth) :param y_pred:is the data predicted (calculated, output) by your model. :return: precision score

def f1_m(y_true, y_pred):

View Source

```
def f1_m(y_true, y_pred):
    """
    Metrics have been removed from Keras core. we need to calculate them manually
    here we will calculate the f1 macro score
    :param y_true:is the true data (or target, ground truth)
    :param y_pred:is the data predicted (calculated, output) by your model.
    :return: f1 macro score
    """
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

Metrics have been removed from Keras core. we need to calculate them manually here we will calculate the f1 macro score :param y_true:is the true data (or target, ground truth) :param y_pred:is the data predicted (calculated, output) by your model. :return: f1 macro score

def CNN(X_train, y_train, X_test, y_test):

View Source

```
def CNN (X_train,y_train ,X_test,y_test):

    """
    Here we have utilized the Convolutional Neural Network deep learning model ,
    we ave used 6 layers with relu as activation function and last
    Dense layer we took sigmoid since we are dealing with multi-label problem.Validation
    setare used to process where a trained model is
```

```
    evaluated with a testing data set.
    binary_accuracy used since we convert our
    multi-label problem to binary.
    :param X_train:Train_Features
    :param y_train:Train_labels
    :param X_test:Test_Features
    :param y_test:Test_labels
    :return:loss, accuracy, f1_score
    """
    input_dim = X_train.shape[1]  # Number of features
    model = Sequential()
    model.add(Dense(5000, input_dim=input_dim, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(600, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(300, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(100, activation="relu"))
    model.add(Dropout(0.5))
    model.add(Dense(60, activation="relu"))
    model.add(Dropout(0.5))
    model.add(Dense(40, activation="relu"))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))

    model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['binary_accuracy',f1_
    model.summary()
    model.fit(X_train, y_train,epochs=20,verbose=True,validation_data=(X_test, y_test),batch

    loss, accuracy, f1_score = model.evaluate(X_test, y_test, verbose=False)
    print("Testing Accuracy:  {:.4f}".format(accuracy))
    print("f1 :  {:.4f}".format(f1_score))
```

Here we have utilized the Convolutional Neural Network deep learning model , we ave used 6 layers with relu as activation function and last Dense layer we took sigmoid since we are dealing with multi-label problem. Validation set are used to process where a trained model is evaluated with a testing data set. binary_accuracy used since we convert our multi-label problem to binary. :param X_train:Train_Features :param y_train:Train_labels :param X_test:Test_Features :param y_test:Test_labels :return:loss, accuracy, f1_score