## 2. Case study dự đoán điểm thi

### Load dữ liệu và nối các tập dữ liệu lại với nhau, sử dụng concat trong pandas

df = pd.concat(map(pd.read\_csv,['./diemthi.csv','./lop71.csv','./lop72.csv']), ignore\_index=True)

df.to\_csv('./diem.csv')

df = pd.read\_csv('./diem.csv')

# xử lý dữ liệu

data = df.fillna(df.mean())

# xóa hàng trùng nhau

cols\_other\_than\_id = list(data.columns)[2:]

print(cols\_other\_than\_id)

data.drop\_duplicates(subset=cols\_other\_than\_id, keep='first', inplace=True)

# lưu lại dữ liệu

data.to\_csv("./data.csv")

# tính toán độ tương quan và vẽ biểu đồ

corr= data.corr()

import seaborn as sns

corr = pd.DataFrame(corr,columns=corr.keys(),index=corr.keys())

fig, ax = plt.subplots(figsize=(15,10))

sns.heatmap(corr, annot=True, cmap='PiYG', linewidths=1, linecolor='white', fmt='.3f',ax=ax)

plt.show()

# select features

X = data.iloc[:, 2:5]

y = data.iloc[:, 5:]

print(X.shape)

print(y.shape)

# chia dữ liệu thành 2 phần train và test

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=42)

# sử dụng linear regression để dự đoán

model = LinearRegression()

model.fit(X\_train,y\_train)

yhat = model.predict(X\_test)

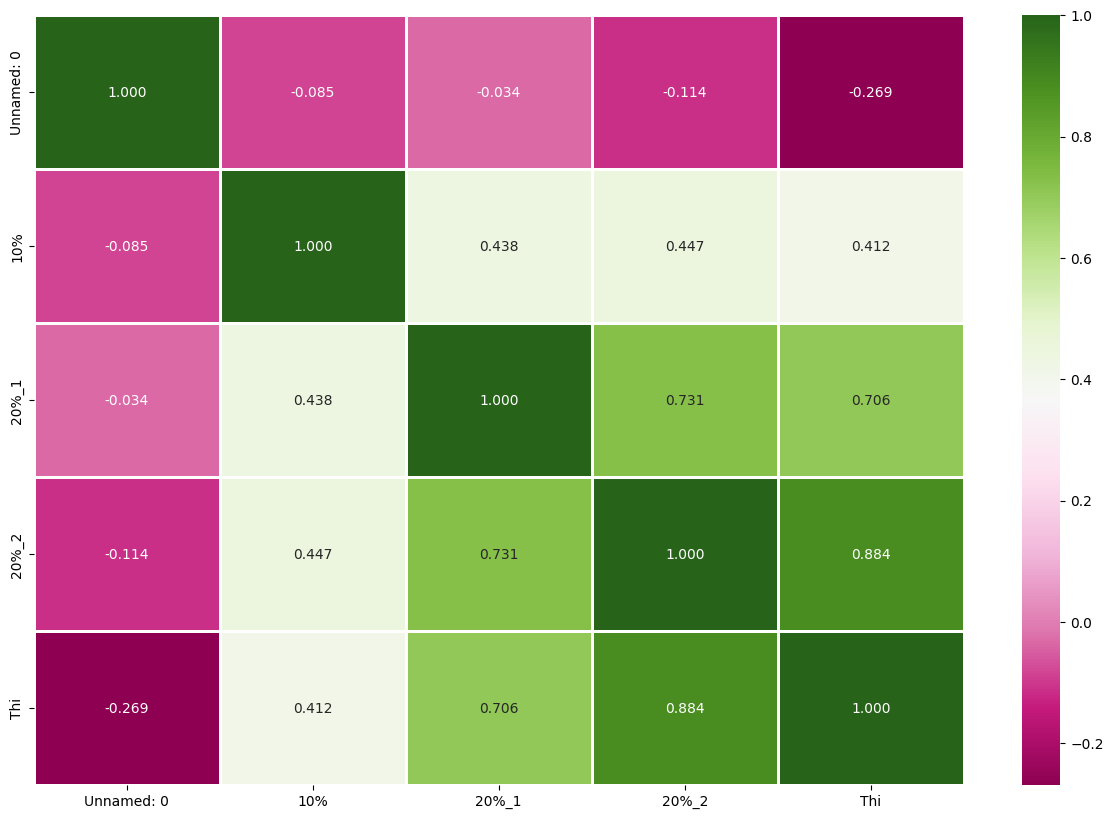
# tính toán độ lệnh và độ chính xác

print(mean\_squared\_error(y\_test,yhat))

print(r2\_score(yhat,y\_test))

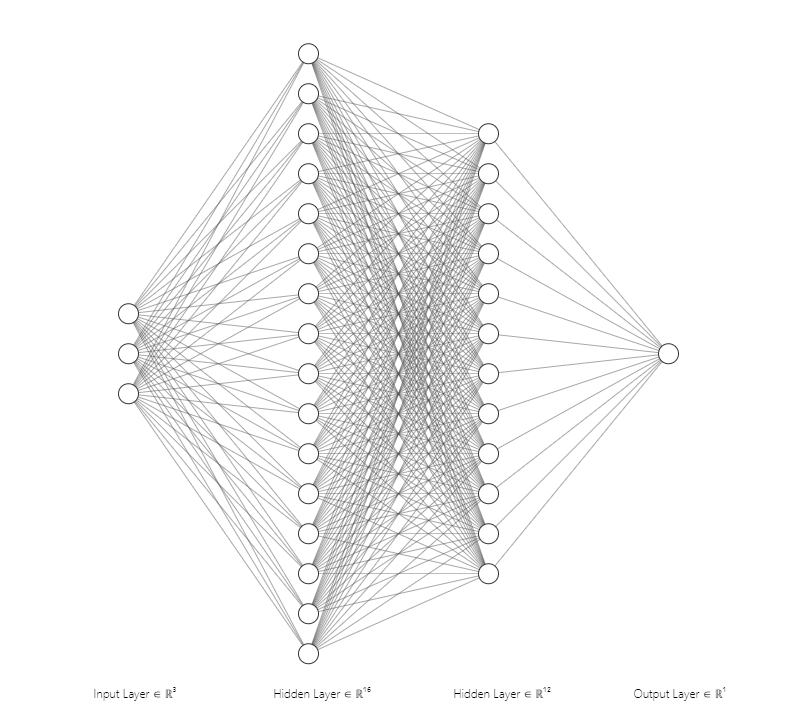
print(pd.DataFrame({'predict':yhat.flatten(),'actual':y\_test['Thi']}))

### # biểu **diễn độ tương quan giữa các thuộc tính**



### sử dụng model deep learning

#### mạng neural sử dụng

****

**Code:**

#model 1

df = pd.read\_csv('./diem.csv')

data = df.fillna(df.mean())

X = data.iloc[:, 2:5]

y = data.iloc[:, 5:]

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=42)

X\_test,X\_val,y\_test,y\_val = train\_test\_split(X\_test,y\_test,test\_size=0.5,random\_state=42)

print(X\_train.shape,y\_train.shape,X\_test.shape,y\_test.shape,X\_val.shape,y\_val.shape)

model = Sequential()

model.add(Dense(16, input\_shape=(3,), activation='relu'))

model.add(Dense(15, activation='relu'))

model.add(Dense(1, activation='linear'))

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=10, batch\_size=32)

plt.plot(model.history.history['loss'], label='loss')

plt.plot(model.history.history['val\_loss'], label='val\_loss')

plt.legend()

plt.show()

plt.plot(model.history.history['mae'], label='mae')

plt.plot(model.history.history['val\_mae'], label='val\_mae')

plt.show()

model.save('model1.h5')

from sklearn.metrics import mean\_squared\_error

# y\_pre = pd.DataFrame(yhat,columns=['predict'])

yhat = model.predict([X\_test])

# print(model.predict([X\_test]))

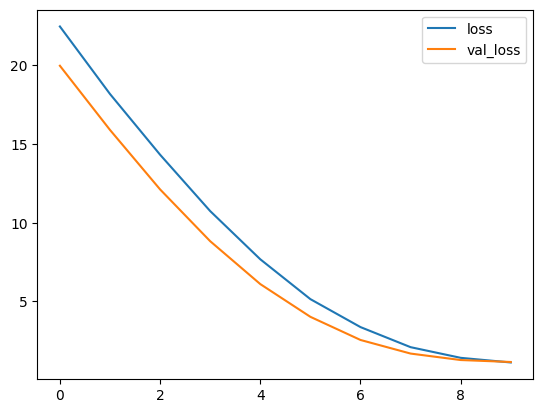
# print(pd.DataFrame({'predict':yhat.flatten(),'actual':y\_test['Thi']}))

print("Độ sai",mean\_squared\_error(y\_test, yhat))

# r2 > 1 kết quả không rõ ràng

from sklearn.metrics import r2\_score

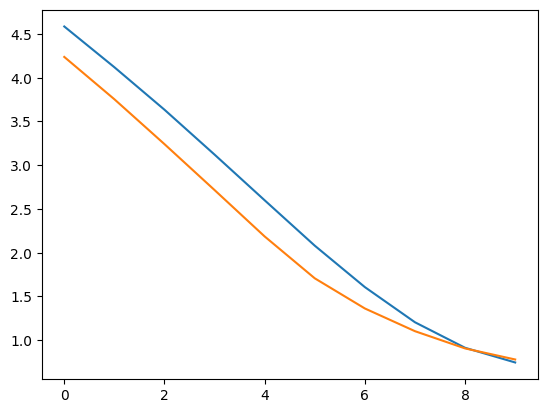
print("Độ tin cậy",r2\_score(y\_test, yhat,force\_finite=False))

****

Loss function là Mse

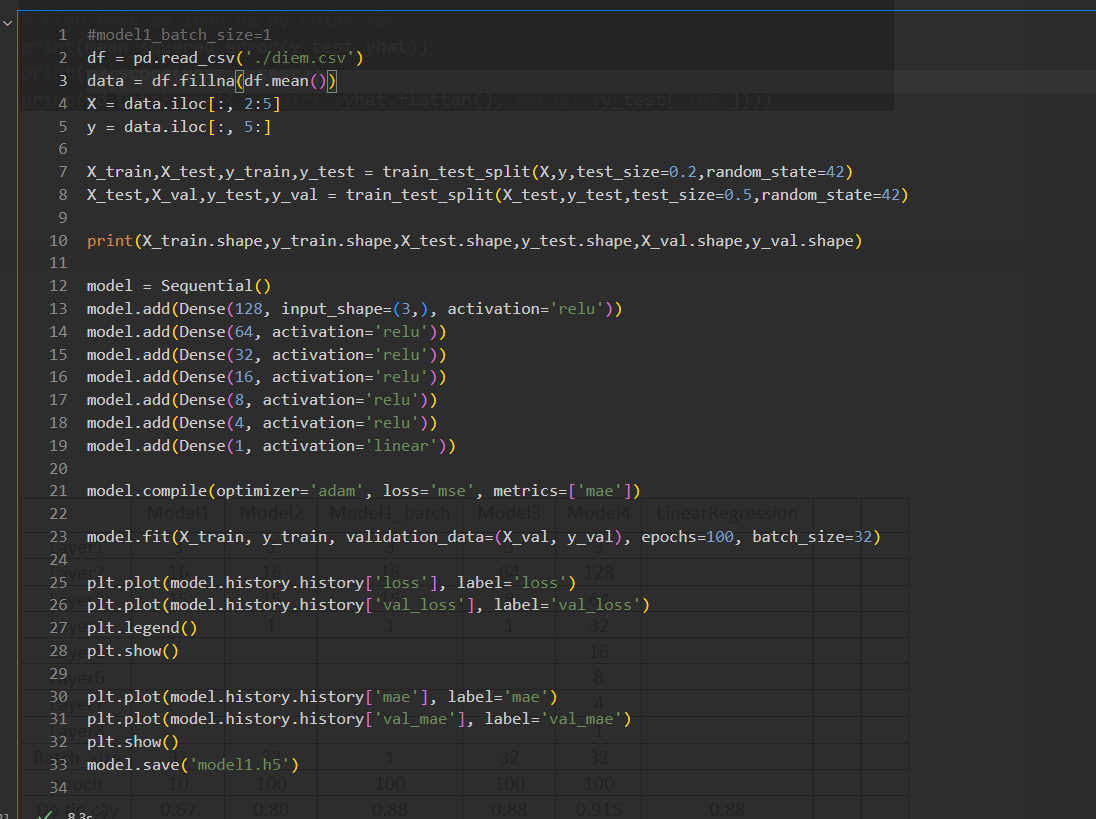


Với epoch thấp

****

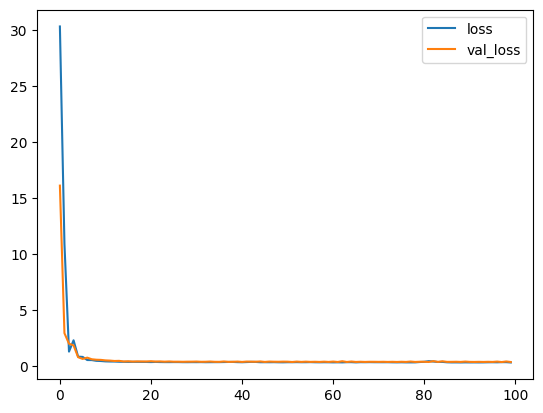
**Metric là mae**

### Chương trình

****

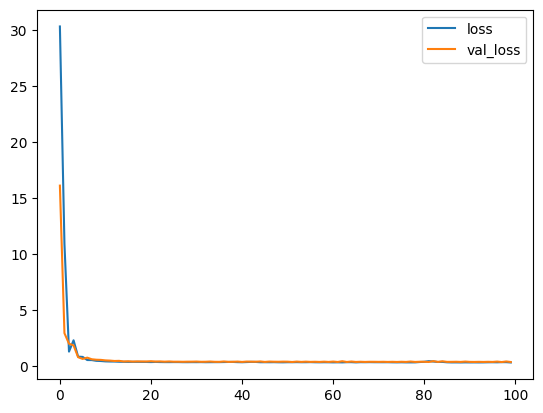
### Biểu đồ loss và metrics

**Loss = mse**

****

**Metrics=mae**

Mean Absolute Error = (1/n) \* ∑|yi – xi|

****

### Thống kê dự đoán điểm thi

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Model1 | Model2 | Model1\_batch | Model3 | Model4 | LinearRegression |  |  |
| Layer1 | 3 | 3 | 3 | 3 | 3 |  |  |  |
| Layer2 | 16 | 16 | 16 | 64 | 128 |  |  |  |
| Layer3 | 15 | 15 | 15 | 8 | 64 |  |  |  |
| Layer4 | 1 | 1 | 1 | 1 | 32 |  |  |  |
| Layer5 |  |  |  |  | 16 |  |  |  |
| Layer6 |  |  |  |  | 8 |  |  |  |
| Layer7 |  |  |  |  | 4 |  |  |  |
| Layer8 |  |  |  |  | 1 |  |  |  |
| Batch\_size | 32 | 32 | 1 | 32 | 32 |  |  |  |
| epoch | 10 | 100 | 100 | 100 | 100 |  |  |  |
| Độ tin cậy | 0.67 | 0.80 | 0.88 | 0.88 | 0.915 | 0.88 |  |  |
| mse | 0.86 | 0.52 | 0.30 | 0.30 | 0.22 | 0.30 |  |  |
|  |  |  |  |  |  |  |  |  |

### Công thức tính toán loss và metrics

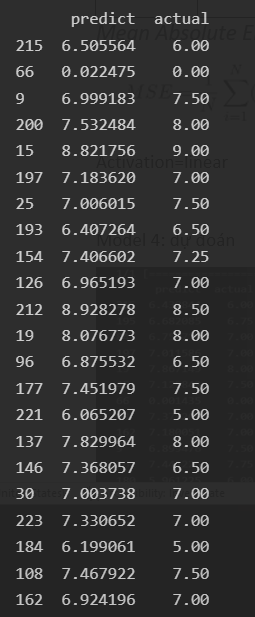
Mean Absolute Error = (1/n) \* ∑|yi – xi|



Activation=linears

Model có sẵn:

### Dự đoán và so sánh với điểm mong muốn



Model 4: dự đoán

