prediction

March 27, 2024

1 Importing modules and reading data

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np

from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score
```

[2]:
```
drug = "AMN-107"
df = pd.read_csv("./drug data/" + drug + ".csv")
```

1.0.1 Preprocessing Data

[3]:
```
x = pd.get_dummies(df.drop(['AUC', 'Cell line'], axis=1))
y = df["AUC"]
```

[4]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.2)
```

2 Adding layers and training the model

[5]:
```
model = Sequential()
```

[6]:
```
model.add(Dense(units=32, activation='relu', input_dim=len(x_train.columns)))
model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))
```

```
D:\anaconda3\envs\TF\Lib\site-packages\keras\src\layers\core\dense.py:88:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

2.0.1 Compiling

[7]:
```
model.compile(loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

2.0.2 Training

[8]:
```
model.fit(x_train,y_train, epochs=50,batch_size=8)
```

```
Epoch 1/50
96/96  2s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.3338
Epoch 2/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.3046
Epoch 3/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.3053
Epoch 4/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2954
Epoch 5/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2786
Epoch 6/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2697
Epoch 44/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2144
Epoch 45/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2115
Epoch 46/50
96/96  1s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2117
Epoch 47/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2060
Epoch 48/50
96/96  0s 3ms/step -
accuracy: 0.0000e+00 - loss: 0.2227
Epoch 49/50
96/96  0s 2ms/step -
accuracy: 0.0000e+00 - loss: 0.2062
Epoch 50/50
96/96  0s 2ms/step -
accuracy: 0.0000e+00 - loss: 0.2204
```

[8]: <keras.src.callbacks.history.History at 0x1b7c2d473b0>

3 Predicition and Finding Correlation

3.0.1 Setting up predictions and test case

[9]:
```
values = model.predict(x_test)

pred = []
for i in values:
    pred.append(i[0])
pred = np.array(pred)
```

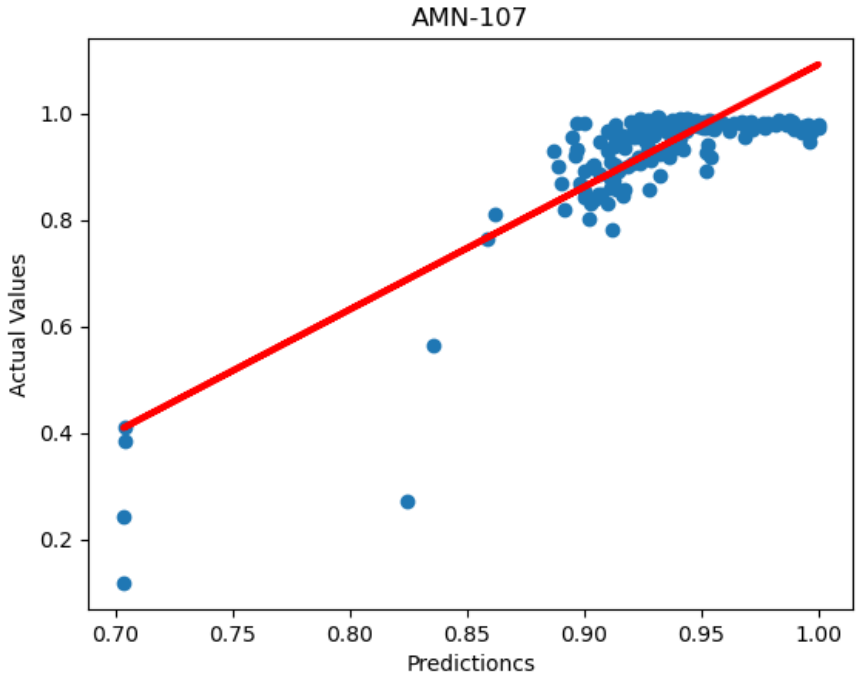
6/6 0s 16ms/step

[10]:
```
y_test = y_test.to_numpy()
```

3.0.2 Plotting Graph

[11]:
```
plt.scatter(pred,y_test)
plt.xlabel("Predictioncs")
plt.ylabel("Actual Values")
plt.title(drug)
m, b = np.polyfit(pred, y_test, 1)
plt.plot(pred, m*pred+b, lw=3, color="red")
```

[11]: [<matplotlib.lines.Line2D at 0x1b7c4112c00>]



4 Finding Pearson correlation coefficient

[12]:
```
import math
```

[13]:
```
pred_n = pred - pred.mean()
y_test_n = y_test - y_test.mean()
```

[14]:
```
num = sum(pred_n*y_test_n)
```

[15]:
```
den = math.sqrt(sum(pred_n**2)*sum(y_test_n**2))
```

PCC	Value
$0 < r \ 0.19$	Very Low Correlation
$0.2 \ r \ 0.39$	Low Correlation
$0.4 \ r \ 0.59$	Moderate Correlation
$0.6 \ r \ 0.79$	High Correlation
$0.8 \ r \ 1.0$	Very High Correlation

[16]:
```
pcc = num/den
print("Pearson correlation coefficient comes out to be:",pcc)
```

```
Pearson correlation coefficient comes out to be: 0.8378189061492276
```