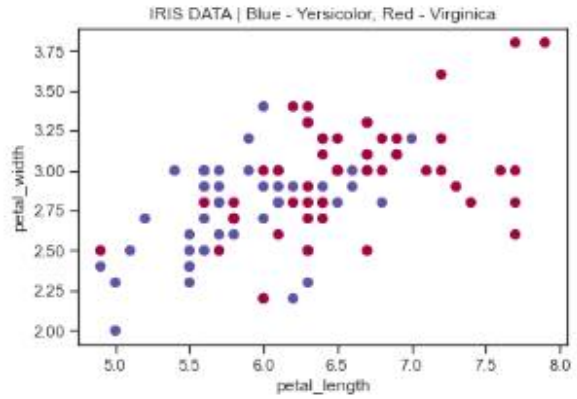
2015039080 정보통신공학부 송상호

케라스를 이용한 붓꽃 분류

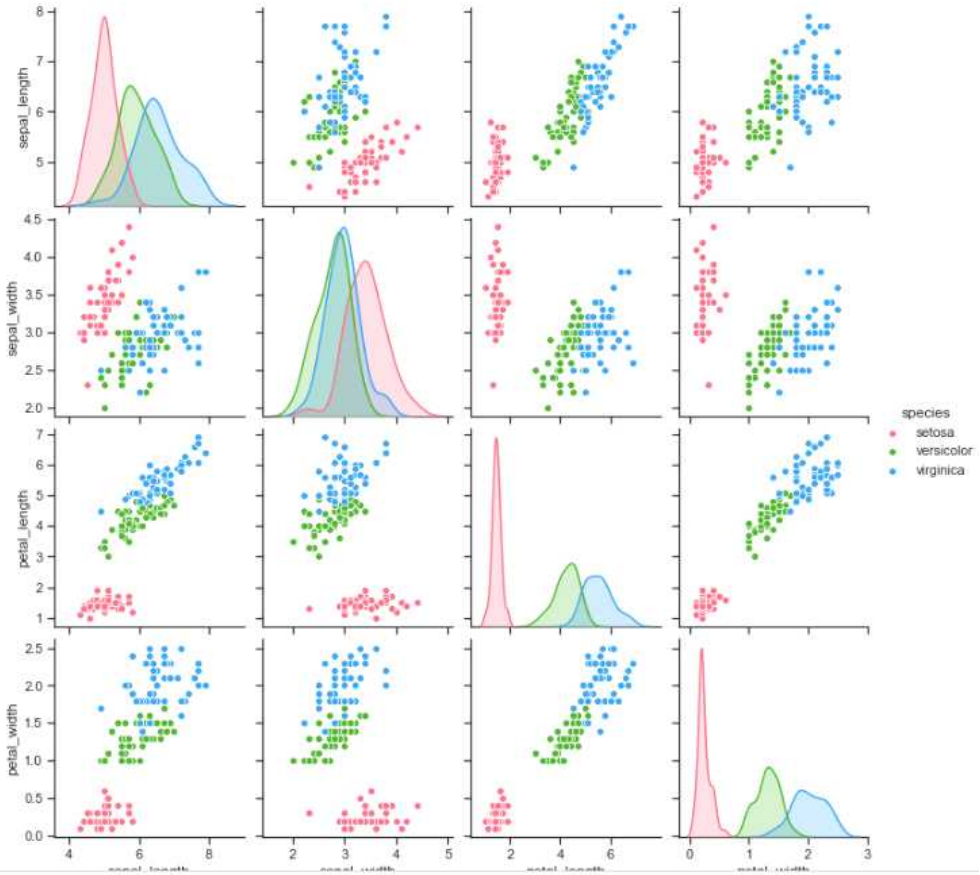
In [30]:
```
#import libraries
import os
import pandas as pd
import matplotlib.pyplot as plt

#Set working directory and load data
os.chdir('C:/PycharmProjects/')
iris =pd.read_csv('iris.csv')

print(iris.head())
#Create numeric classes for species(0,1,2)
iris.loc[iris['species']=='virginica','species']=0
iris.loc[iris['species']=='versicolor','species']=1
iris.loc[iris['species']=='setosa','species']=2
iris = iris[iris['species']!=2]

#Create input and output colums
X = iris[['sepal_length','sepal_width']].values.T
Y = iris[['species']].values.T
Y=Y.astype('uint8')

#Make a scatter plot
plt.scatter(X[0,:], X[1,:],c=Y[0,:],s=40, cmap=plt.cm.Spectral);
plt.title("IRIS DATA | Blue - Versicolor, Red - Virginica")
plt.xlabel('petal_length')
plt.ylabel('petal_width')
plt.show()
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```



In [21]:

```
import seaborn as sns
import pandas as pd
import numpy as np

sns.set(style="ticks", color_codes=True)
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species", palette="husl")
```



In [22]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [23]:
```
iris['species'].unique()
```

Out[23]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [24]:
```
from sklearn.preprocessing import LabelEncoder

X = iris.iloc[:,0:4].values
y = iris.iloc[:,4].values

encoder =  LabelEncoder()
y1 = encoder.fit_transform(y)
Y = pd.get_dummies(y1).values
Y
```

```
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       [0, 0, 1]], dtype=uint8)
```

In [25]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y,
                                                    test_size=0.2,
                                                    random_state=1)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[25]:
```
((120, 4), (30, 4), (120, 3), (30, 3))
```

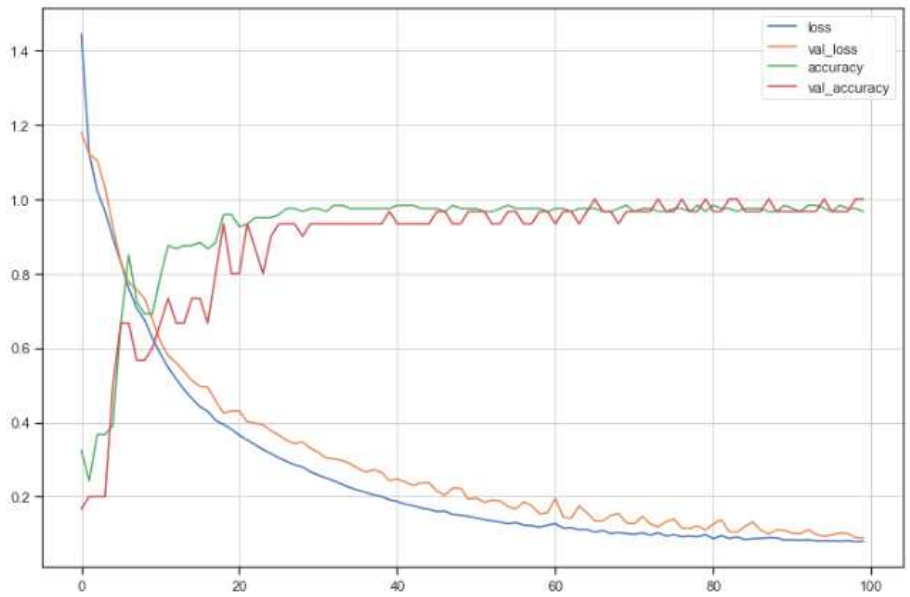
In [26]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

model = Sequential()

model.add(Dense(64,input_shape=(4,),activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(3,activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 64)                320
_________________________________________________________________
dense_5 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 195
=================================================================
Total params: 4,675
Trainable params: 4,675
Non-trainable params: 0
_________________________________________________________________
```

In [27]:
```
hist = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100)
```

```
120/120 [==============================] - 0s 67us/step - loss: 0.6747 - accuracy: 0.6917 - val_loss: 0.7334 - val_accurac
y: 0.5667
Epoch 10/100
120/120 [==============================] - 0s 67us/step - loss: 0.6255 - accuracy: 0.6917 - val_loss: 0.6759 - val_accurac
y: 0.6000
Epoch 11/100
120/120 [==============================] - 0s 142us/step - loss: 0.5841 - accuracy: 0.7917 - val_loss: 0.6167 - val_accura
cy: 0.6667
Epoch 12/100
120/120 [==============================] - 0s 83us/step - loss: 0.5469 - accuracy: 0.8750 - val_loss: 0.5789 - val_accurac
y: 0.7333
Epoch 13/100
120/120 [==============================] - 0s 83us/step - loss: 0.5164 - accuracy: 0.8667 - val_loss: 0.5600 - val_accurac
y: 0.6667
Epoch 14/100
120/120 [==============================] - 0s 92us/step - loss: 0.4875 - accuracy: 0.8750 - val_loss: 0.5368 - val_accurac
y: 0.6667
Epoch 15/100
120/120 [==============================] - 0s 75us/step - loss: 0.4637 - accuracy: 0.8750 - val_loss: 0.5116 - val_accurac
y: 0.7333
```

In [29]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,8))
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.legend(['loss','val_loss', 'accuracy','val_accuracy'])
plt.grid()
plt.show()
```



In [ ]: