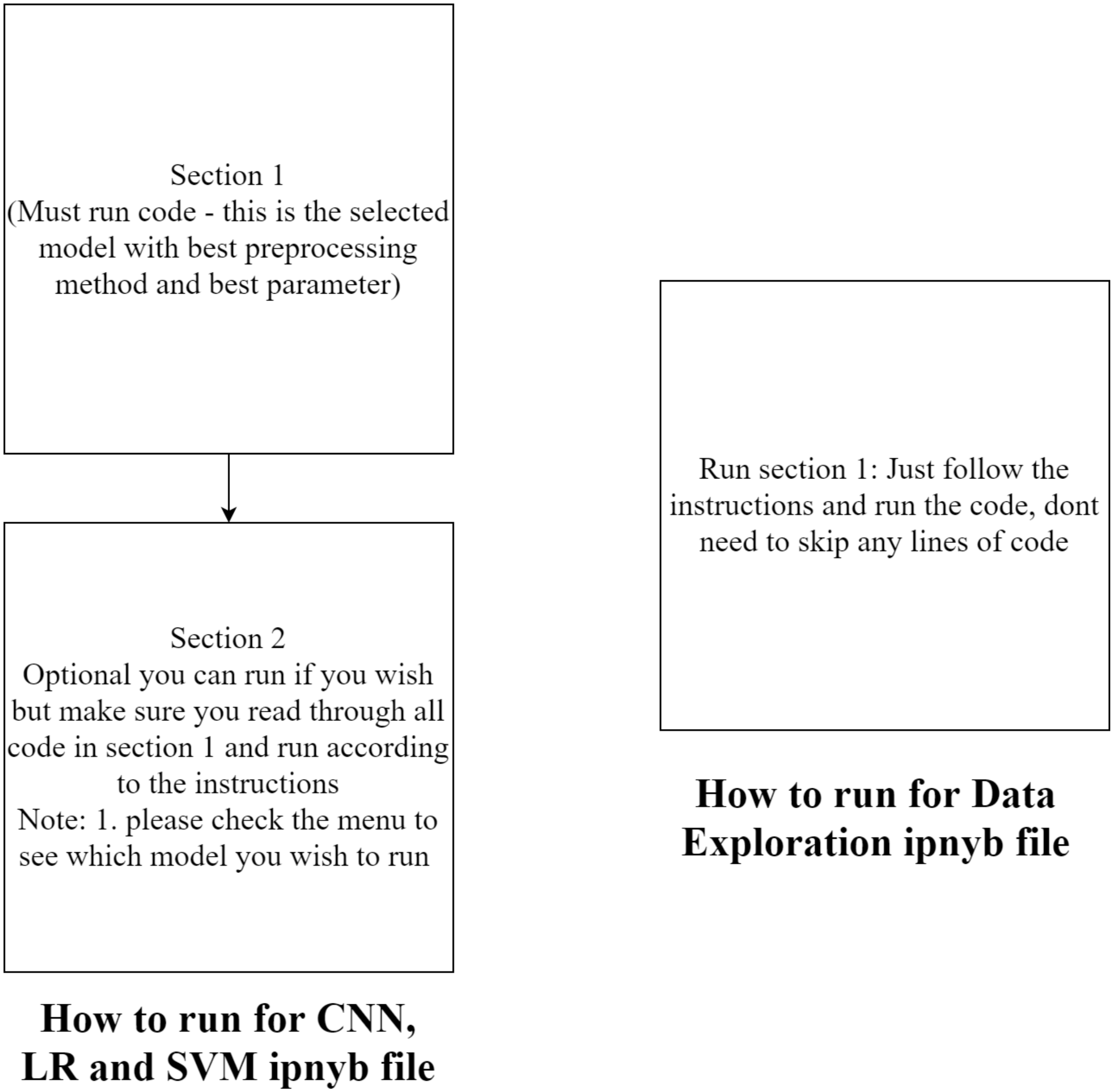

DATA EXPLORATION

SECTION 1

just run the following code and stop at the part where you wish to stop

```
#libraries
```

```
from keras.datasets import cifar100
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPool2D
from keras.layers.core import Dense,Activation,Dropout,Flatten
from keras.utils import np_utils
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow.keras import datasets, layers, models, optimizers
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from statsmodels.formula.api import glm
import keras

# Download dataset of CIFAR-100
(x_train,y_train),(x_test,y_test) = cifar100.load_data()

#print shape of the dataset
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
print('y_train shape:', y_train.shape)
print('y_test shape:', y_test.shape)

# print number of data set samples
print(x_train.shape[0], 'train set')
print(x_test.shape[0], 'test set')

# Data type for train and test set
print(type(x_test))
print(type(y_test[0]))
```

```
x_train shape: (50000, 32, 32, 3)
x_test shape: (10000, 32, 32, 3)
y_train shape: (50000, 1)
y_test shape: (10000, 1)
50000 train set
10000 test set
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

CIFAR-100 Labels

0: apple

1: aquarium_fish

2: baby

3: bear

4: beaver

5: bed

6: bee

7: beetle

8: bicycle

9: bottle

10: bowl

11: boy

12: bridge

13: bus

14: butterfly

15: camel

16: can

17: castle

18: caterpillar

19: cattle

20: chair

21: chimpanzee

22: clock

23: cloud

24: cockroach

25: couch

26: cra

27: crocodile

28: cup

29: dinosaur

30: dolphin

31: elephant

32: flatfish

33: forest

34: fox

35: girl

36: hamster

37: house

38: kangaroo

39: keyboard

40: lamp

41: lawn_mower

42: leopard

43: lion

44: lizard

45: lobster

46: man

47: maple_tree

48: motorcycle

49: mountain

50: mouse

51: mushroom

52: oak_tree

53: orange

54: orchid

55: otter

56: palm_tree

57: pear

58: pickup_truck

59: pine_tree

60: plain

61: plate

62: poppy

63: porcupine

64: possum

65: rabbit

66: raccoon

67: ray

68: road

69: rocket

70: rose

71: sea

72: seal

73: shark

74: shrew

75: skunk

76: skyscraper

77: snail

78: snake

79: spider

80: squirrel

81: streetcar

82: sunflower

83: sweet_pepper

84: table

85: tank

86: telephone

87: television

88: tiger

89: tractor

90: train

91: trout

92: tulip

93: turtle

94: wardrobe

95: whale

96: willow_tree

97: wolf

98: woman

99: worm

visualise some images

```
# Show sample random image 5x5
plt.figure(figsize=(10,10))
for i in range(25):
    rand_num=np.random.randint(0,100)
    cifar_image=plt.subplot(5,5,i+1)
    plt.imshow(x_train[rand_num])
    # Erase the value of x tick and y tick
    plt.xticks(color="None")
    plt.yticks(color="None")
    # remove the tick x-axis and y-axis
    plt.tick_params(length=0)
    # print label
    plt.title(y_train[rand_num])

plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\text.py:1165: FutureWarning: eleme
  if s != self._text:
```


```
%%time
#flipping image

#preprocess dataset
Datagenerator = ImageDataGenerator()

x_train1 = Datagenerator.apply_transform(x=x_train, transform_parameters={'flip_horizontal':T
x_test1 = Datagenerator.apply_transform(x=x_test, transform_parameters={'flip_horizontal':Tru

#add back to original input image
x_train = np.concatenate((x_train,x_train1))
x_test = np.concatenate((x_test,x_test1))
#append the label twice because the fiiped images is the same images just being flipped, it h
y_train = np.concatenate((y_train,y_train))
y_test = np.concatenate((y_test,y_test))
```

```
Wall time: 460 ms
```

```
%%time
# Normalize taining and test set image to the range of 0-1
x_train = x_train.astype('float32')/255.0
x_test = x_test.astype('float32')/255.0

# convert the labels of y_train,y_test to One-Hot encoding
y_train = np_utils.to_categorical(y_train,100)
y_test = np_utils.to_categorical(y_test,100)
```

```
Wall time: 3.76 s
```

```
#split train into train and validation and keep test dataset as it is
x_train, x_valid, y_train , y_valid = train_test_split(x_train, y_train,
                                                        train_size=0.70,
                                                        random_state=42,
                                                        stratify=y_train)
```

we learned how to construct the following code from https://towardsdatascience.com/assumptions-of-logistic-regression-clearly-explained-44d85a22b290

```
#create another set of train, valid and test to get the get the label that are not in one hot

(x_train100,y_train100),(x_test100,y_test100) = cifar100.load_data()


y_train100 = np.concatenate((y_train100,y_train100))
x_train100 = np.concatenate((x_train100,x_train100))
x_test100 = np.concatenate((x_test100,x_test100))
y_test100 = np.concatenate((y_test100,y_test100))

from sklearn.model_selection import train_test_split
x_train100, x_valid100, y_train100 , y_valid100 = train_test_split(x_train100, y_train100,
                                                        train_size=0.70,
                                                        random_state=42,
                                                        stratify=y_train100)
```

running Box-Tidwell Test to test linearity assumption

```
# Add constant term
xt = x_train.reshape(x_train.shape[0],x_train.shape[1]*x_train.shape[2]*x_train.shape[3])
X_lt = sm.add_constant(xt, prepend=False)

# Building model and fit the data (using statsmodel's Logit)
logit_results = sm.GLM(y_train100, X_lt, family=sm.families.Poisson()).fit()

# Display summary results
print(logit_results.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                70000
Model:                            GLM   Df Residuals:                    66927
Model Family:                 Poisson   Df Model:                         3072
Link Function:                    log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:            -8.4078e+05
Date:                Wed, 27 Oct 2021   Deviance:                   1.3022e+06
```

```
Time:                        12:13:29   Pearson chi2:                 1.13e+06
No. Iterations:                     5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.2702      0.050     -5.442      0.000      -0.368      -0.173
x2             0.2916      0.060      4.872      0.000       0.174       0.409
x3             0.0366      0.043      0.849      0.396      -0.048       0.121
x4             0.7708      0.071     10.917      0.000       0.632       0.909
x5            -0.9618      0.082    -11.673      0.000      -1.123      -0.800
x6             0.1928      0.063      3.084      0.002       0.070       0.315
x7            -0.6792      0.073     -9.337      0.000      -0.822      -0.537
x8             0.9508      0.085     11.187      0.000       0.784       1.117
x9            -0.1334      0.065     -2.049      0.040      -0.261      -0.006
x10            0.7473      0.074     10.151      0.000       0.603       0.892
x11           -0.9163      0.087    -10.592      0.000      -1.086      -0.747
x12            0.0421      0.066      0.637      0.524      -0.088       0.172
x13           -0.5523      0.076     -7.242      0.000      -0.702      -0.403
x14            0.8106      0.088      9.160      0.000       0.637       0.984
x15           -0.2046      0.067     -3.050      0.002      -0.336      -0.073
x16           -0.0151      0.077     -0.195      0.845      -0.167       0.136
x17           -0.2155      0.090     -2.386      0.017      -0.393      -0.038
x18            0.2535      0.068      3.749      0.000       0.121       0.386
x19            0.0867      0.078      1.117      0.264      -0.065       0.239
x20            0.1257      0.091      1.388      0.165      -0.052       0.303
x21           -0.2557      0.068     -3.786      0.000      -0.388      -0.123
x22           -0.0637      0.075     -0.854      0.393      -0.210       0.082
x23           -0.2465      0.088     -2.789      0.005      -0.420      -0.073
x24            0.2671      0.066      4.047      0.000       0.138       0.397
x25            0.2777      0.074      3.778      0.000       0.134       0.422
x26            0.3217      0.087      3.688      0.000       0.151       0.493
x27           -0.4441      0.066     -6.745      0.000      -0.573      -0.315
x28           -0.3370      0.073     -4.629      0.000      -0.480      -0.194
x29           -0.2056      0.085     -2.428      0.015      -0.372      -0.040
x30            0.5127      0.065      7.937      0.000       0.386       0.639
x31            0.3097      0.072      4.307      0.000       0.169       0.451
x32            0.0277      0.084      0.330      0.742      -0.137       0.192
x33           -0.3321      0.064     -5.187      0.000      -0.458      -0.207
x34           -0.0209      0.069     -0.303      0.762      -0.156       0.114
x35           -0.2585      0.081     -3.191      0.001      -0.417      -0.100
x36            0.2529      0.061      4.149      0.000       0.133       0.372
x37           -0.3347      0.068     -4.926      0.000      -0.468      -0.202
x38            0.3476      0.082      4.263      0.000       0.188       0.507
x39            0.0472      0.061      0.771      0.441      -0.073       0.167
x40            0.4479      0.069      6.474      0.000       0.312       0.584
x41           -0.2058      0.082     -2.496      0.013      -0.367      -0.044
x42           -0.2741      0.062     -4.397      0.000      -0.396      -0.152
x43           -0.1329      0.070     -1.911      0.056      -0.269       0.003
x44           -0.3669      0.083     -4.435      0.000      -0.529      -0.205
x45            0.3789      0.063      6.030      0.000       0.256       0.502
```

Code learned from website: https://towardsdatascience.com/how-to-create-fast-and-accurate-scatter-plots-with-lots-of-data-in-python-a1d3f578e551

```
#plot scatter plot input data map with output labels
```

```
#plot scatter plot input data map with output labels
(x_train1, y_train1), (x_test1, y_test1) = keras.datasets.cifar100.load_data()
assert x_train1.shape == (50000, 32, 32, 3)
assert x_test1.shape == (10000, 32, 32, 3)
assert y_train1.shape == (50000, 1)
assert y_test1.shape == (10000, 1)

plt.figure()
train = x_train1.reshape(x_train1.shape[0],x_train1.shape[1]*x_train1.shape[2]*x_train1.shape
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
proj = pca.fit_transform(train)
plt.scatter(proj[:, 0], proj[:, 1], c=y_train1, cmap="Paired")
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x17f5b54a4c0>
```