In [28]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

file_data   = "mnist.csv"
handle_file = open(file_data, "r")
data        = handle_file.readlines()
handle_file.close()


size_row    = 28    # height of the image
size_col    = 28    # width of the image

num_image   = len(data)
count       = 0     # count for the number of images


    #
# normalize the values of the input data to be [0, 1]
#
def normalize(data):

    data_normalized = (data - min(data)) / (max(data) - min(data))

    return(data_normalized)

#
# make a matrix each column of which represents an images in a vector form
#
list_image  = np.empty((size_row * size_col, num_image), dtype=float)
list_label  = np.empty(num_image, dtype=int)

for line in data:

    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])
    im_vector   = normalize(im_vector)

    list_label[count]       = label
    list_image[:, count]    = im_vector

    count += 1


y, x= list_label, list_image
x = x.transpose()


def plot_history(net_history):
    history = net_history.history
    import matplotlib.pyplot as plt
    losses = history['loss']
    val_losses = history['val_loss']
    accuracies = history['accuracy']
    val_accuracies = history['val_accuracy']
```

```
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.plot(losses, 'r')
    plt.plot(val_losses, 'b')
    plt.legend(['loss', 'val_loss'])

    plt.figure()
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.plot(accuracies, 'b')
    plt.plot(val_accuracies, 'r')
    plt.legend(['acc', 'val_acc'])

# Load data
train_images,test_images, train_labels, test_labels= train_test_split(x,y,train_s


# Data attributes
print("train_images dimentions: ", train_images.ndim)
print("train_images shape: ", train_images.shape)
print("train_images type: ", train_images.dtype)

X_train = train_images.reshape(5000, 784)
X_test = test_images.reshape(5000, 784)

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_test /= 255

from keras.utils import np_utils
Y_train = np_utils.to_categorical(train_labels)
Y_test = np_utils.to_categorical(test_labels)

#=================================================
# Creating our model
from keras.models import Sequential
from keras.layers import Dense, Dropout,Conv2D
from keras.optimizers import SGD
from keras.losses import categorical_crossentropy

myModel = Sequential()
myModel.add(Dense(196, activation='relu', input_shape=(784,)))
myModel.add(Dropout(20))
myModel.add(Dense(49, activation='relu'))
myModel.add(Dropout(20))
myModel.add(Dense(10, activation='softmax'))


myModel.summary()
myModel.compile(optimizer=SGD(lr=0.001), loss=categorical_crossentropy, metrics=[


#=================================================
# Train our model
```

```
network_history = myModel.fit(X_train, Y_train, batch_size=128, epochs=30, valida
score = myModel.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
plot_history(network_history)

# Evaluation
test_loss, test_acc = myModel.evaluate(X_test, Y_test)
test_labels_p = myModel.predict(X_test)
import numpy as np
test_labels_p = np.argmax(test_labels_p, axis=1)

# Change layers config
myModel.layers[0].name = 'Layer_0'
myModel.layers[0].trainable = False
myModel.layers[0].get_config()
```

```
train_images dimentions:  2
train_images shape:  (5000, 784)
train_images type:  float64
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_66 (Dense)             (None, 196)               153860
_________________________________________________________________
dropout_18 (Dropout)         (None, 196)               0
_________________________________________________________________
dense_67 (Dense)             (None, 49)                9653
_________________________________________________________________
dropout_19 (Dropout)         (None, 49)                0
_________________________________________________________________
dense_68 (Dense)             (None, 10)                500
=================================================================
Total params: 164,013
Trainable params: 164,013
```

In [24]:
```
x.shape
```

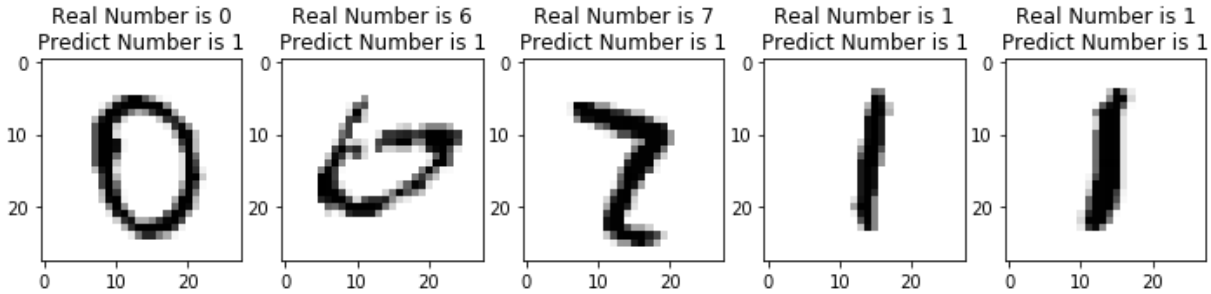
Out[24]: (10000, 784)

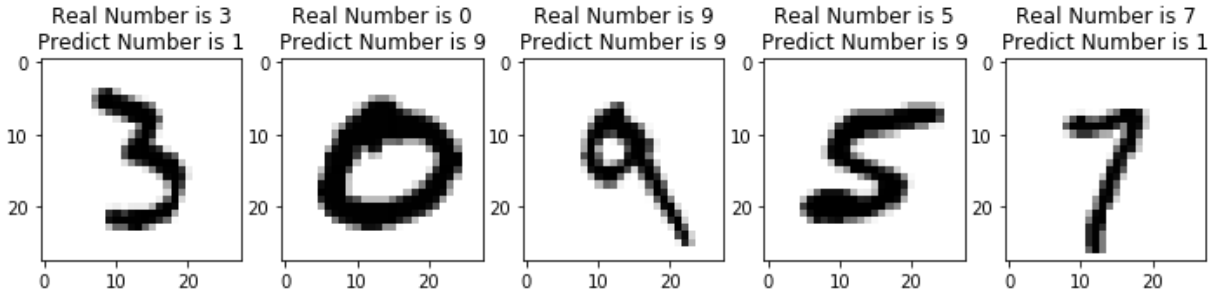
In [25]:
```
y.shape
```

Out[25]: (10000,)

In [61]:
```
y_pred = myModel.predict(X_test)
X_test__ = X_test.reshape(X_test.shape[0], 28, 28)

fig, axis = plt.subplots(2, 5, figsize=(12, 14))
for i, ax in enumerate(axis.flat):
    ax.imshow(X_test__[i], cmap='binary')
    ax.set(title = f"Real Number is {Y_test[i].argmax()}\nPredict Number is {y_pr
```





In [ ]:

In [ ]: