```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Load Dataset

```
train_df = pd.read_csv('fashion-mnist_train.csv')
test_df = pd.read_csv('fashion-mnist_test.csv')
```

```
train_df.shape
```

```
(1887, 785)
```

```
test_df.shape
```

```
(2832, 785)
```

```
train_df.describe()
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	1
count	1887.000000	1887.000000	1887.000000	1887.000000	1887.000000	1887.000000	
mean	4.366190	0.007419	0.011659	0.035506	0.129306	0.240064	
std	2.845432	0.322286	0.279887	0.355700	2.356873	3.387508	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	7.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
max	9.000000	14.000000	10.000000	10.000000	87.000000	88.000000	

8 rows × 785 columns

```
train_df.label.unique()
```

```
array([2, 9, 6, 0, 3, 4, 5, 8, 7, 1])
```

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat','Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'
```

Preprocess Data

Convert each image of 784 into (28x28x1)(height x width x color_channels). Divide values by 255 to scale the values.

```
x_train = train_df.iloc[:,1:].to_numpy()
x_train = x_train.reshape([-1,28,28,1])
x_train = x_train / 255
```

```
y_train = train_df.iloc[:,0].to_numpy()
```

```
x_test = test_df.iloc[:,1:].to_numpy()
x_test = x_test.reshape([-1,28,28,1])
x_test = x_test / 255
```

```
y_test = test_df.iloc[:,0].to_numpy()
```

Visualization

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```

Model Building

```
from keras.models import Sequential
from keras.layers import Dense,Conv2D,Flatten,MaxPooling2D,Dropout
```

```
model = Sequential()

model.add(Conv2D(filters=64,kernel_size=(3,3),input_shape=(28,28,1),activation='relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(rate=0.3))
model.add(Flatten())
model.add(Dense(units=32, activation='relu'))
```

```
model.add(Dense(units=10, activation='sigmoid'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 dropout (Dropout)           (None, 13, 13, 64)        0

 flatten (Flatten)           (None, 10816)             0

 dense (Dense)               (None, 32)                346144

 dense_1 (Dense)             (None, 10)                330

=================================================================
Total params: 347,114
Trainable params: 347,114
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(x_train,y_train,epochs=50,batch_size=1200,validation_split=0.05)
```

```
Epoch 23/50
2/2 [==============================] - 2s 566ms/step - loss: 0.4670 - accuracy: 0.8343 - val_loss: nan - val_a
Epoch 24/50
2/2 [==============================] - 2s 585ms/step - loss: 0.4568 - accuracy: 0.8382 - val_loss: nan - val_a
```

```
Epoch 47/50
2/2 [==============================] - 2s 588ms/step - loss: 0.3060 - accuracy: 0.8945 - val_loss: nan - val_a
Epoch 48/50
2/2 [==============================] - 2s 583ms/step - loss: 0.3021 - accuracy: 0.8934 - val_loss: nan - val_a
Epoch 49/50
2/2 [==============================] - 2s 577ms/step - loss: 0.3001 - accuracy: 0.8968 - val_loss: nan - val_a
Epoch 50/50
2/2 [==============================] - 2s 587ms/step - loss: 0.3005 - accuracy: 0.8934 - val_loss: nan - val_a
<keras.callbacks.History at 0x7f2c97f92920>
```

Evaluation

```
evaluation = model.evaluate(x_test,y_test)
```

```
89/89 [==============================] - 1s 8ms/step - loss: nan - accuracy: 0.8422
```

```
print(f"Accuracy: {evaluation[1]}")
```

```
Accuracy: 0.8421609997749329
```

```
y_probas = model.predict(x_test)
```

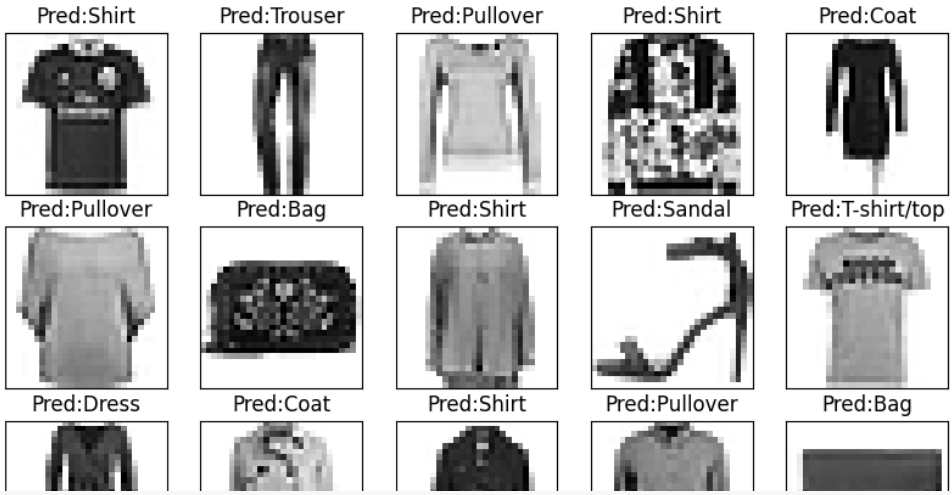
```
89/89 [==============================] - 1s 14ms/step
```

```
y_pred = y_probas.argmax(axis=-1)
```

```
y_pred
```

```
array([6, 1, 2, ..., 2, 9, 0])
```

```
plt.figure(figsize=(10,10),)
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_test[i], cmap=plt.cm.binary)
#     plt.xlabel(f"True Class:{y_test[i]}")
    plt.title(f"Pred:{class_names[y_pred[i]]}")
plt.show()
```




```
from sklearn.metrics import classification_report
```

```
num_classes = 10
class_names = ["class {}".format(i) for i in range(num_classes)]
cr = classification_report(y_test, y_pred, target_names=class_names)
print(cr)
```

```
                precision    recall  f1-score   support

       class 0       0.81      0.74      0.77       278
       class 1       0.99      0.95      0.97       266
       class 2       0.74      0.74      0.74       281
       class 3       0.86      0.90      0.88       301
       class 4       0.74      0.80      0.77       280
       class 5       0.94      0.91      0.93       289
       class 6       0.62      0.61      0.62       293
       class 7       0.88      0.87      0.87       264
       class 8       0.95      0.95      0.95       281
       class 9       0.90      0.95      0.92       299

      accuracy                           0.84      2832
     macro avg       0.84      0.84      0.84      2832
  weighted avg       0.84      0.84      0.84      2832
```