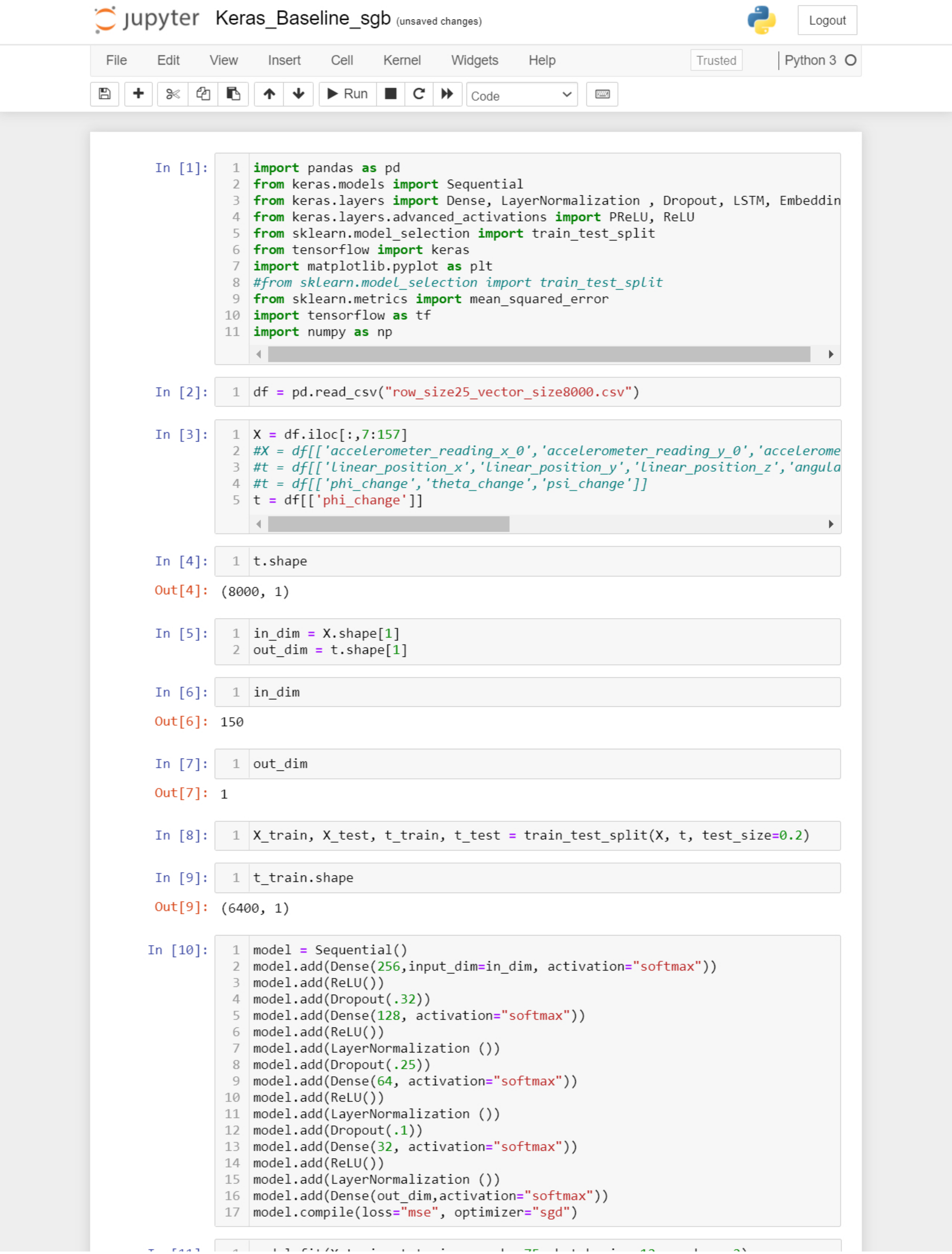


In [1]:
```python
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, LayerNormalization , Dropout, LSTM, Embeddin
from keras.layers.advanced_activations import PReLU, ReLU
from sklearn.model_selection import train_test_split
from tensorflow import keras
import matplotlib.pyplot as plt
#from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import tensorflow as tf
import numpy as np
```

In [2]:
```python
df = pd.read_csv("row_size25_vector_size8000.csv")
```

In [3]:
```python
X = df.iloc[:,7:157]
#X = df[['accelerometer_reading_x_0','accelerometer_reading_y_0','accelerome
#t = df[['linear_position_x','linear_position_y','linear_position_z','angula
#t = df[['phi_change','theta_change','psi_change']]
t = df[['phi_change']]
```

In [4]:
```python
t.shape
```

Out[4]:
```
(8000, 1)
```

In [5]:
```python
in_dim = X.shape[1]
out_dim = t.shape[1]
```

In [6]:
```python
in_dim
```

Out[6]:
```
150
```

In [7]:
```python
out_dim
```

Out[7]:
```
1
```

In [8]:
```python
X_train, X_test, t_train, t_test = train_test_split(X, t, test_size=0.2)
```

In [9]:
```python
t_train.shape
```

Out[9]:
```
(6400, 1)
```

In [10]:
```python
model = Sequential()
model.add(Dense(256,input_dim=in_dim, activation="softmax"))
model.add(ReLU())
model.add(Dropout(.32))
model.add(Dense(128, activation="softmax"))
model.add(ReLU())
model.add(LayerNormalization ())
model.add(Dropout(.25))
model.add(Dense(64, activation="softmax"))
model.add(ReLU())
model.add(LayerNormalization ())
model.add(Dropout(.1))
model.add(Dense(32, activation="softmax"))
model.add(ReLU())
model.add(LayerNormalization ())
model.add(Dense(out_dim,activation="softmax"))
model.compile(loss="mse", optimizer="sgd")
```

In [11]:
```python
model.fit(X_train, t_train, epochs=75, batch_size=12, verbose=2)
```

In [11]:
```
model.fit(X_train, t_train, epochs=75, batch_size=12, verbose=2)
```

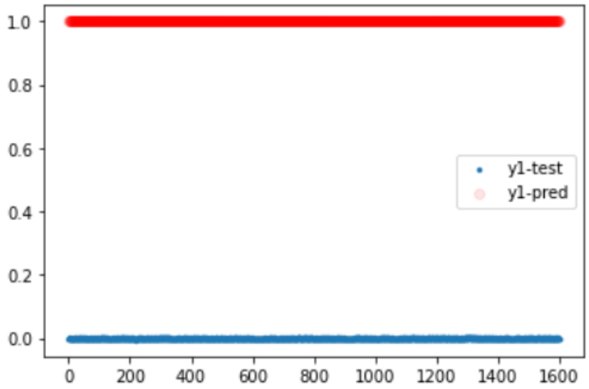
```
534/534 - 1s - loss: 1.0000
Epoch 67/75
534/534 - 1s - loss: 1.0000
Epoch 68/75
534/534 - 1s - loss: 1.0000
Epoch 69/75
534/534 - 1s - loss: 1.0000
Epoch 70/75
534/534 - 1s - loss: 1.0000
Epoch 71/75
534/534 - 1s - loss: 1.0000
Epoch 72/75
534/534 - 1s - loss: 1.0000
Epoch 73/75
534/534 - 1s - loss: 1.0000
Epoch 74/75
534/534 - 1s - loss: 1.0000
Epoch 75/75
534/534 - 1s - loss: 1.0000
```

Out[11]:
```
<tensorflow.python.keras.callbacks.History at 0x1b80486b850>
```

In [12]:
```
ypred = model.predict(X_test)
print("y1 MSE: ", mean_squared_error(t_test.iloc[:, 0], ypred[:,0]))
#print("y2 MSE: ", mean_squared_error(t_test.iloc[:, 1], ypred[:,1]))
#print("y3 MSE: ", mean_squared_error(t_test.iloc[:, 2], ypred[:,2]))

```

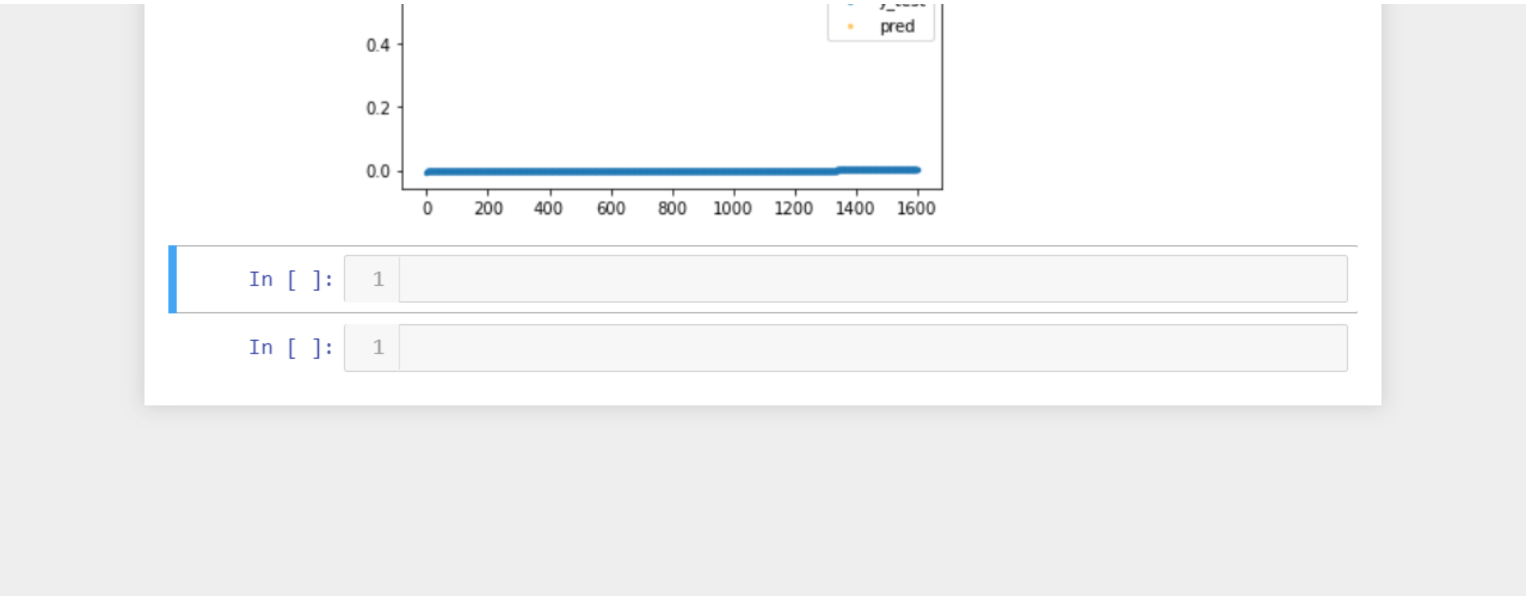
```
y1 MSE:  1.0000413298495425
```

In [13]:
```
x_ax = range(len(X_test))

plt.scatter(x_ax, t_test.iloc[:, 0],  s=6, label="y1-test")
plt.scatter(x_ax, ypred[:,0], label="y1-pred",c="red",alpha = 0.1)

plt.legend()
plt.show()
```



In [14]:
```
x_ax = range(len(X_test))

y_test_index = np.argsort(t_test.iloc[:, 0], axis=0).to_numpy()

f = plt.figure()
plt.scatter(x_ax, t_test.iloc[y_test_index],  s=6, label="y_test")
plt.scatter(x_ax, ypred[y_test_index], s=6, label="pred",c="orange", alpha=0
#plt.ylim(t_test.iloc[y_test_index[0]].to_numpy()[0])
plt.legend()
plt.show()

f.savefig("foo.pdf", bbox_inches='tight')
```

1.0
0.8
0.6
y_test


y_test
pred
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400
1600
In [ ]:
In [ ]: