Deep Learning Italia

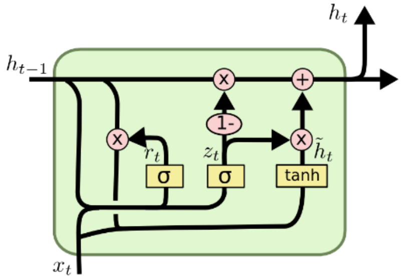
Speaker: Matteo Alberti Deep Learning Consultant @ Techedge | Community ML @ DeepLearningItalia

Email: *m.alberti@deeplearninglitalia.com*

Applicazioni di Reti Neurali Profonde nel mondo Finance:

- Apprendimento Supervisionato
  - Convoluzioni (CNN)
  - Reti Ricorrenti (LSTM)
  - Conv1D + LSTM
- Apprendimento Non Supervisionato
  - Da PCA ad AutoEncoder (V-AE)
- Apprendimento per Rinforzo
  - Automated Stock-Trading

Reti Neurali Ricorrenti - LSTM

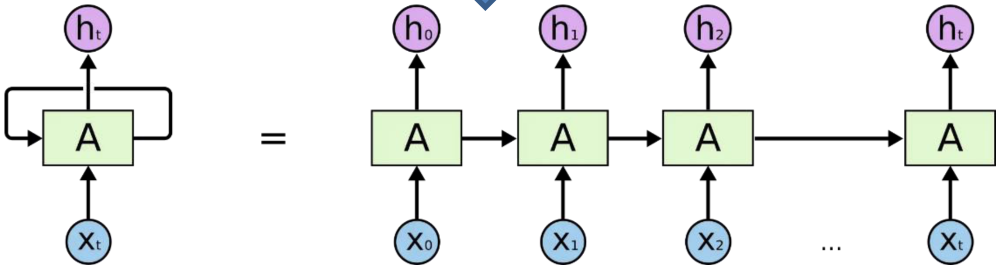


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



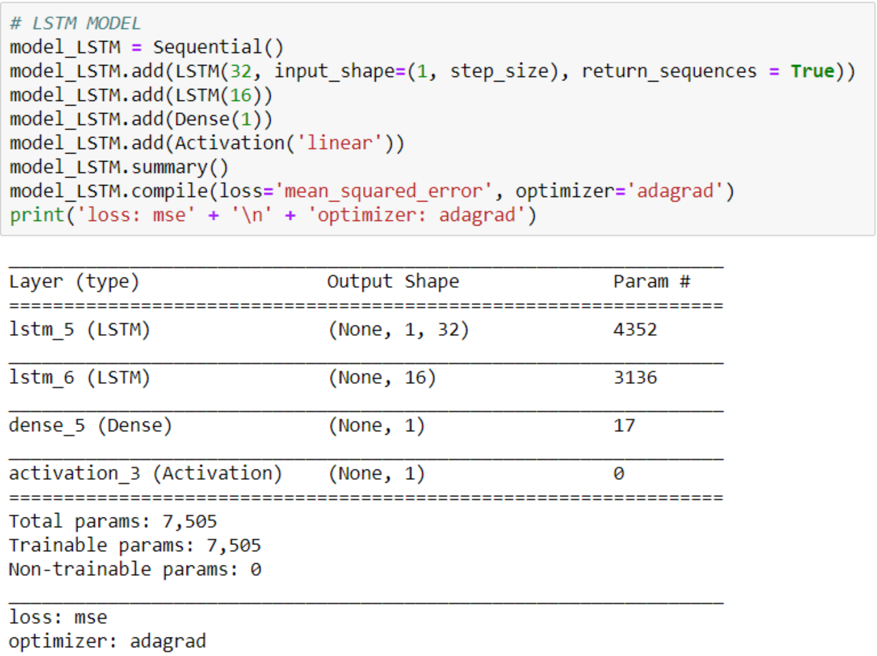
Short-Term Dependencies

- *Le reti LSTM & GRU fanno parte delle «reti ricorrenti»*

Long-Term Dependencies

- Utilizziamo reti LSTM per sequence modelling: con le serie storiche si utilizzano «reti ricorrenti»

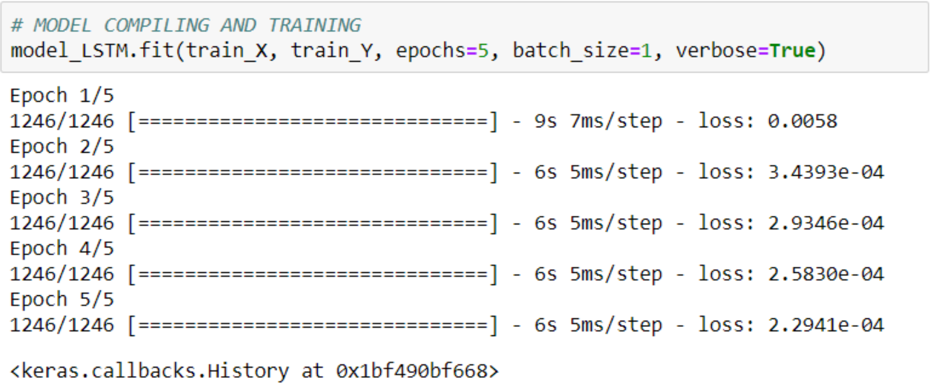
Build Model



```
# LSTM MODEL
model_LSTM = Sequential()
model_LSTM.add(LSTM(32, input_shape=(1, step_size), return_sequences = True))
model_LSTM.add(LSTM(16))
model_LSTM.add(Dense(1))
model_LSTM.add(Activation('linear'))
model_LSTM.summary()
model_LSTM.compile(loss='mean_squared_error', optimizer='adagrad')
print('loss: mse' + '\n' + 'optimizer: adagrad')
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_5 (LSTM)                (None, 1, 32)             4352      
_________________________________________________________________
lstm_6 (LSTM)                (None, 16)                3136      
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 17        
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0         
=================================================================
Total params: 7,505
Trainable params: 7,505
Non-trainable params: 0
_________________________________________________________________
loss: mse
optimizer: adagrad
```

Fit Model

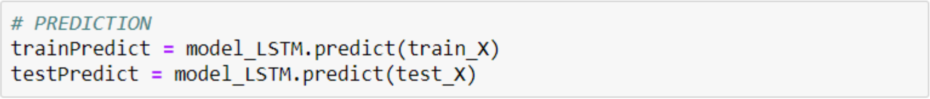


```
# MODEL COMPILING AND TRAINING
model_LSTM.fit(train_X, train_Y, epochs=5, batch_size=1, verbose=True)
```

```
Epoch 1/5
1246/1246 [==============================] - 9s 7ms/step - loss: 0.0058
Epoch 2/5
1246/1246 [==============================] - 6s 5ms/step - loss: 3.4393e-04
Epoch 3/5
1246/1246 [==============================] - 6s 5ms/step - loss: 2.9346e-04
Epoch 4/5
1246/1246 [==============================] - 6s 5ms/step - loss: 2.5830e-04
Epoch 5/5
1246/1246 [==============================] - 6s 5ms/step - loss: 2.2941e-04

<keras.callbacks.History at 0x1bf490bf668>
```

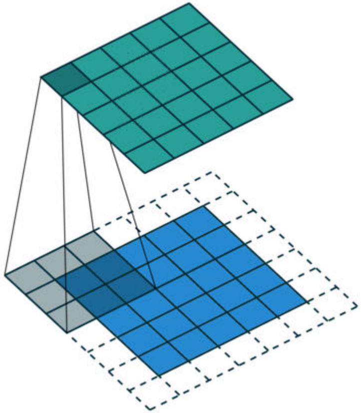
Predict



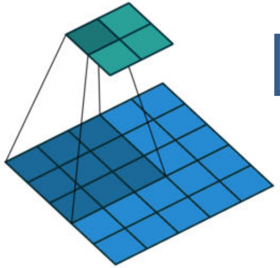
```
# PREDICTION
trainPredict = model_LSTM.predict(train_X)
testPredict = model_LSTM.predict(test_X)
```

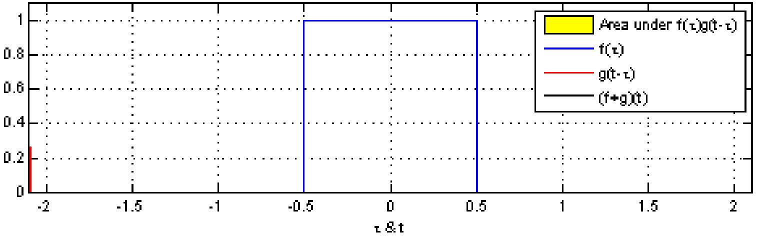
Operatore di Convoluzione

❏ convoluzione

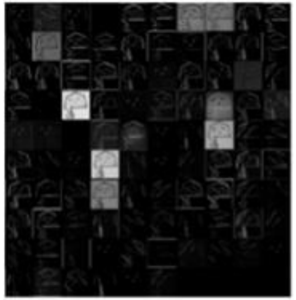


❏ Pooling

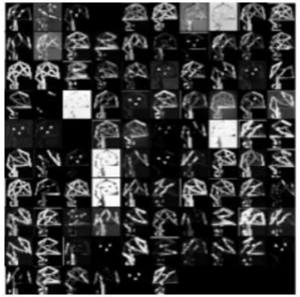




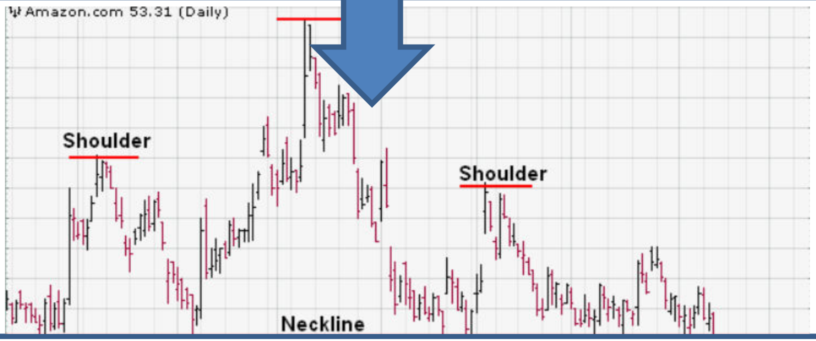
Input



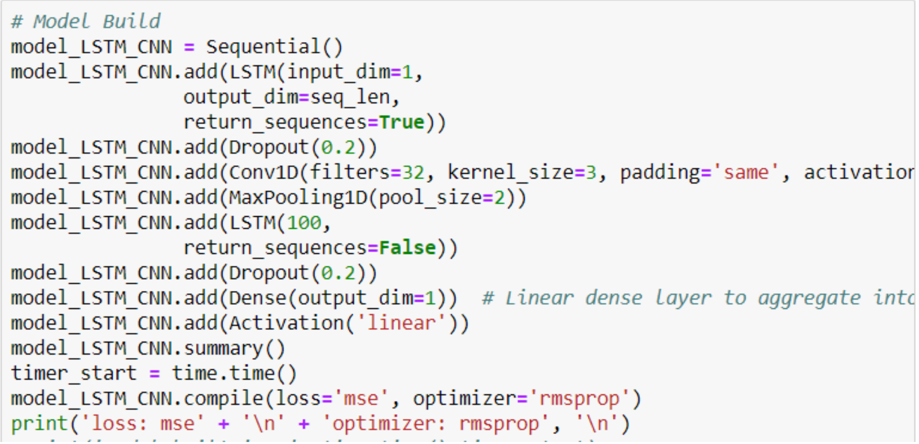
Convoluzione



Pooling

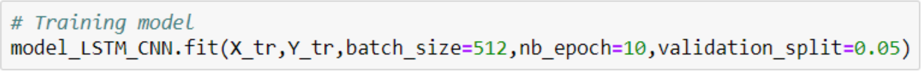


Build Model

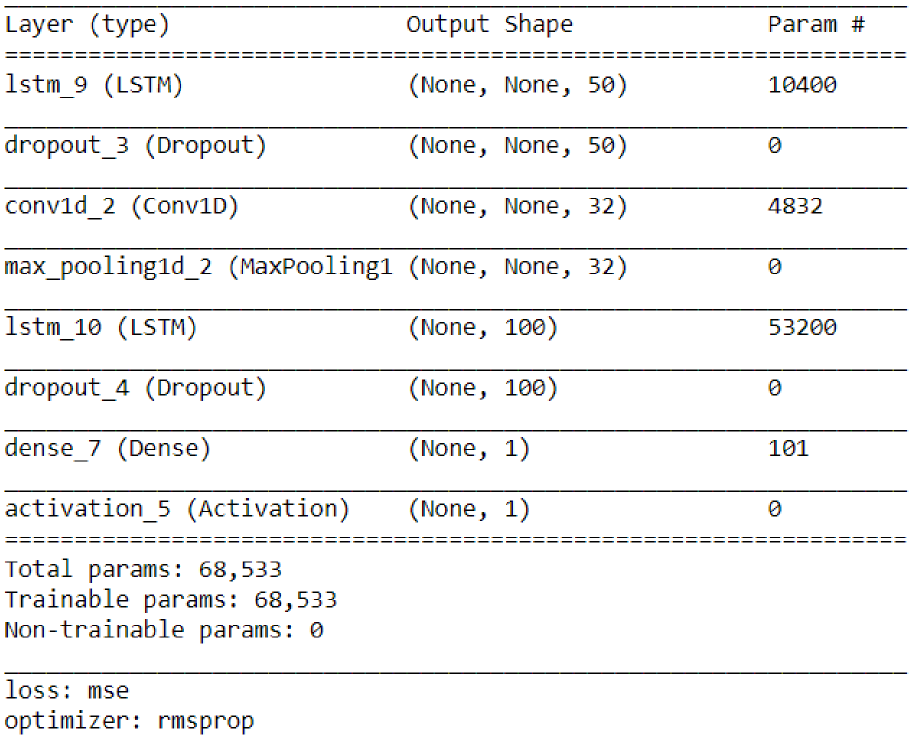


```
# Model Build
model_LSTM_CNN = Sequential()
model_LSTM_CNN.add(LSTM(input_dim=1,
               output_dim=seq_len,
               return_sequences=True))
model_LSTM_CNN.add(Dropout(0.2))
model_LSTM_CNN.add(Conv1D(filters=32, kernel_size=3, padding='same', activation
model_LSTM_CNN.add(MaxPooling1D(pool_size=2))
model_LSTM_CNN.add(LSTM(100,
               return_sequences=False))
model_LSTM_CNN.add(Dropout(0.2))
model_LSTM_CNN.add(Dense(output_dim=1))  # Linear dense layer to aggregate into
model_LSTM_CNN.add(Activation('linear'))
model_LSTM_CNN.summary()
timer_start = time.time()
model_LSTM_CNN.compile(loss='mse', optimizer='rmsprop')
print('loss: mse' + '\n' + 'optimizer: rmsprop', '\n')
```

Fit Model

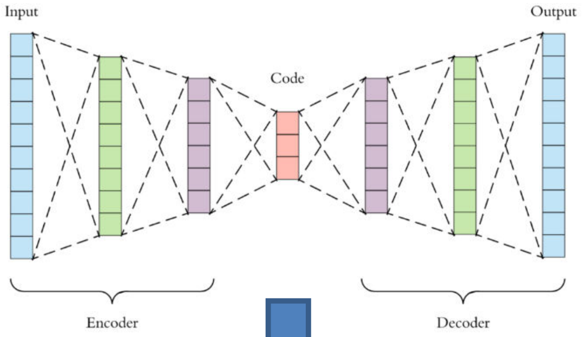


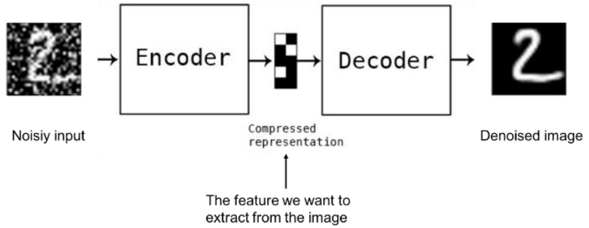
```
# Training model
model_LSTM_CNN.fit(X_tr,Y_tr,batch_size=512,nb_epoch=10,validation_split=0.05)
```

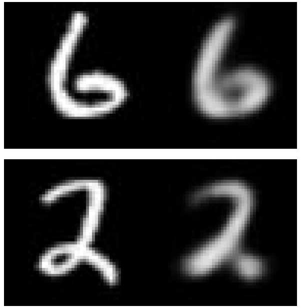


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, None, 50)          10400
_________________________________________________________________
dropout_3 (Dropout)          (None, None, 50)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, None, 32)          4832
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, None, 32)          0
_________________________________________________________________
lstm_10 (LSTM)               (None, 100)               53200
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 101
_________________________________________________________________
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 68,533
Trainable params: 68,533
Non-trainable params: 0
_________________________________________________________________
loss: mse
optimizer: rmsprop
```

Unsupervised Learning - Autoencoder

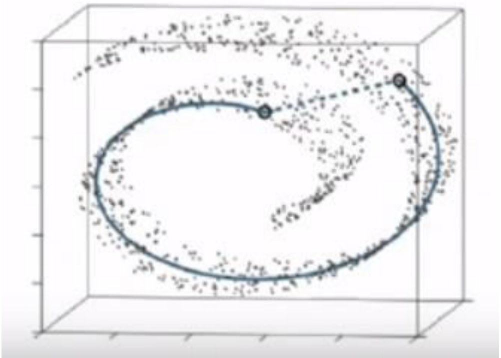



Encoder
Decoder
Noisiy input
Compressed representation
Denoised image
The feature we want to extract from the image



Un passo indietro..

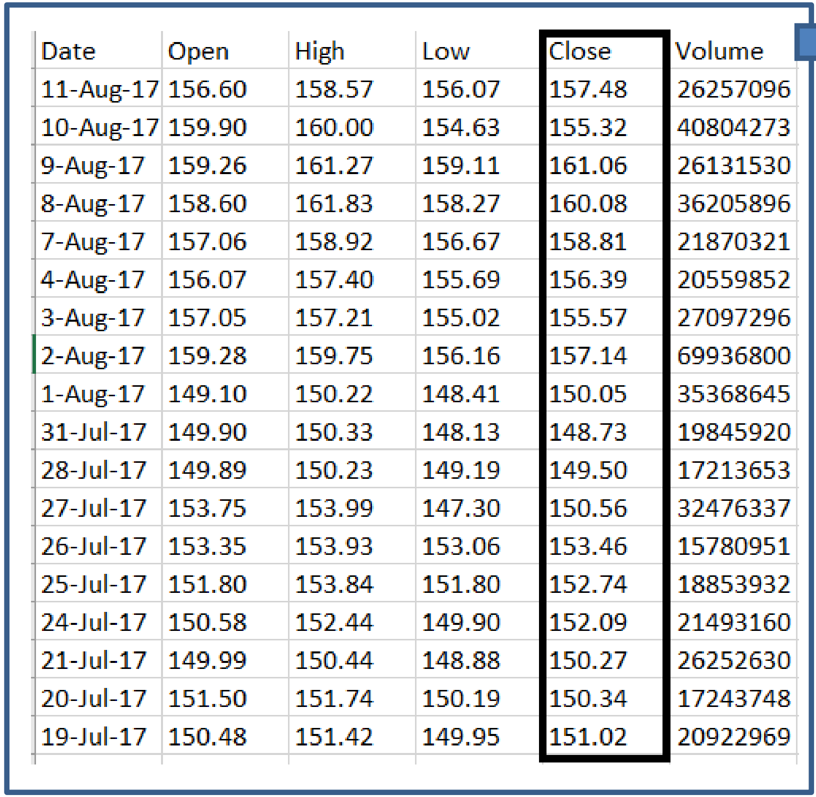
Principal Component Analysis (PCA)



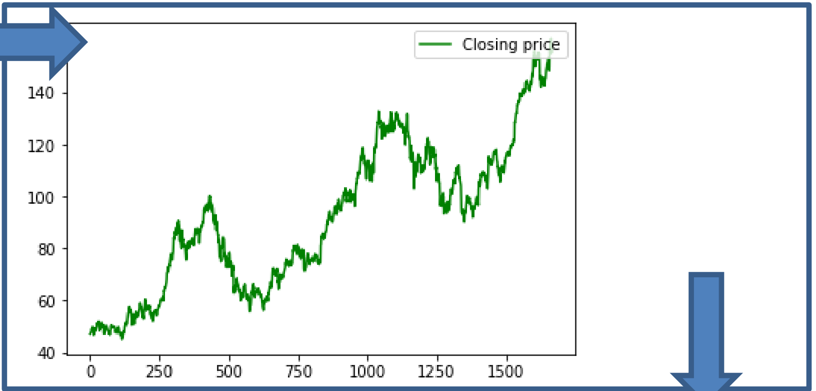
Ma:

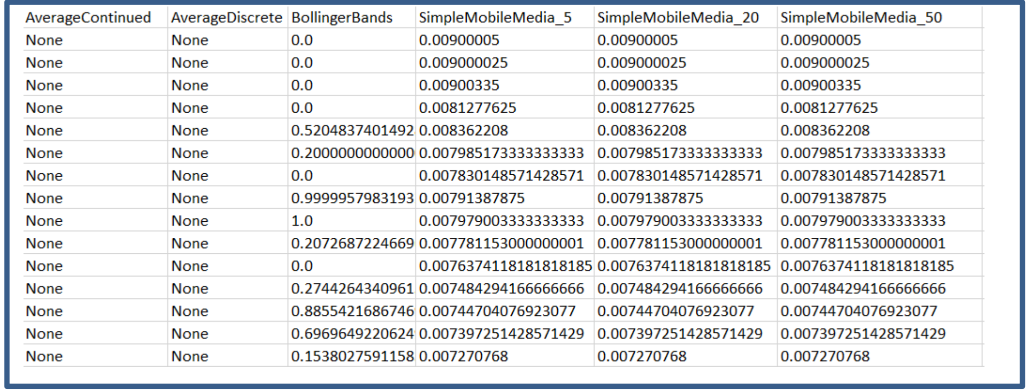
- □ *Possiamo Salvare i pesi*
- □ *Riduzione in sottospazi non lineari*

Unsupervised Learning – Autoencoder Parte 2



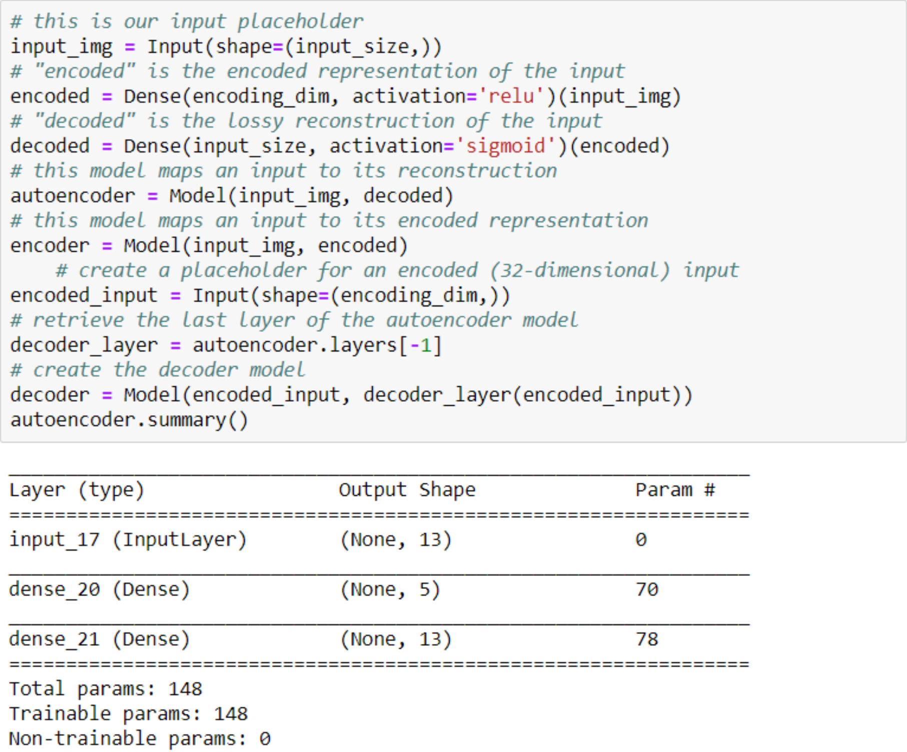
Date	Open	High	Low	Close	Volume
11-Aug-17	156.60	158.57	156.07	157.48	26257096
10-Aug-17	159.90	160.00	154.63	155.32	40804273
9-Aug-17	159.26	161.27	159.11	161.06	26131530
8-Aug-17	158.60	161.83	158.27	160.08	36205896
7-Aug-17	157.06	158.92	156.67	158.81	21870321
4-Aug-17	156.07	157.40	155.69	156.39	20559852
3-Aug-17	157.05	157.21	155.02	155.57	27097296
2-Aug-17	159.28	159.75	156.16	157.14	69936800
1-Aug-17	149.10	150.22	148.41	150.05	35368645
31-Jul-17	149.90	150.33	148.13	148.73	19845920
28-Jul-17	149.89	150.23	149.19	149.50	17213653
27-Jul-17	153.75	153.99	147.30	150.56	32476337
26-Jul-17	153.35	153.93	153.06	153.46	15780951
25-Jul-17	151.80	153.84	151.80	152.74	18853932
24-Jul-17	150.58	152.44	149.90	152.09	21493160
21-Jul-17	149.99	150.44	148.88	150.27	26252630
20-Jul-17	151.50	151.74	150.19	150.34	17243748
19-Jul-17	150.48	151.42	149.95	151.02	20922969





AverageContinued	AverageDiscrete	BollingerBands	SimpleMobileMedia_5	SimpleMobileMedia_20	SimpleMobileMedia_50
None	None	0.0	0.00900005	0.00900005	0.00900005
None	None	0.0	0.009000025	0.009000025	0.009000025
None	None	0.0	0.00900335	0.00900335	0.00900335
None	None	0.0	0.0081277625	0.0081277625	0.0081277625
None	None	0.5204837401492	0.008362208	0.008362208	0.008362208
None	None	0.2000000000000	0.00798517333333333	0.00798517333333333	0.00798517333333333
None	None	0.0	0.00783014857142857 1	0.00783014857142857 1	0.00783014857142857 1
None	None	0.9999957983193	0.00791387875	0.00791387875	0.00791387875
None	None	1.0	0.00797900333333333	0.00797900333333333	0.00797900333333333
None	None	0.2072687224669	0.00778115300000000 1	0.00778115300000000 1	0.00778115300000000 1
None	None	0.0	0.00763741181818185	0.00763741181818185	0.00763741181818185
None	None	0.2744264340961	0.00748429416666666	0.00748429416666666	0.00748429416666666
None	None	0.8855421686746	0.0074470407692307 7	0.0074470407692307 7	0.0074470407692307 7
None	None	0.6969649220624	0.00739725142857142 9	0.00739725142857142 9	0.00739725142857142 9
None	None	0.1538027591158	0.007270768	0.007270768	0.007270768

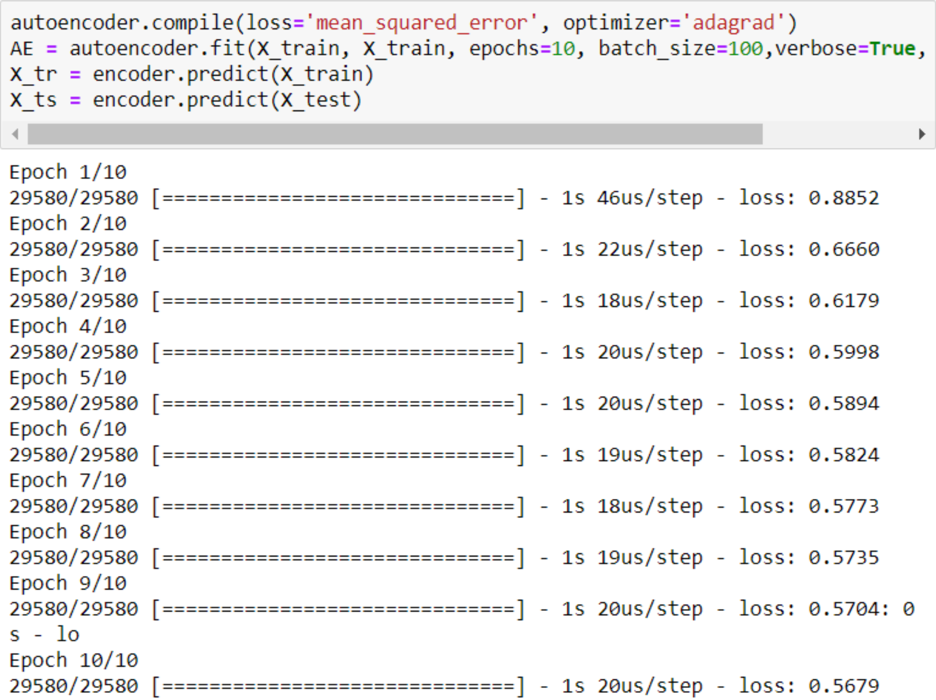
Build Model



```
# this is our input placeholder
input_img = Input(shape=(input_size,))
# "encoded" is the encoded representation of the input
encoded = Dense(encoding_dim, activation='relu')(input_img)
# "decoded" is the lossy reconstruction of the input
decoded = Dense(input_size, activation='sigmoid')(encoded)
# this model maps an input to its reconstruction
autoencoder = Model(input_img, decoded)
# this model maps an input to its encoded representation
encoder = Model(input_img, encoded)
    # create a placeholder for an encoded (32-dimensional) input
encoded_input = Input(shape=(encoding_dim,))
# retrieve the last layer of the autoencoder model
decoder_layer = autoencoder.layers[-1]
# create the decoder model
decoder = Model(encoded_input, decoder_layer(encoded_input))
autoencoder.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_17 (InputLayer)        (None, 13)                0         
_________________________________________________________________
dense_20 (Dense)             (None, 5)                 70        
_________________________________________________________________
dense_21 (Dense)             (None, 13)                78        
=================================================================
Total params: 148
Trainable params: 148
Non-trainable params: 0
```

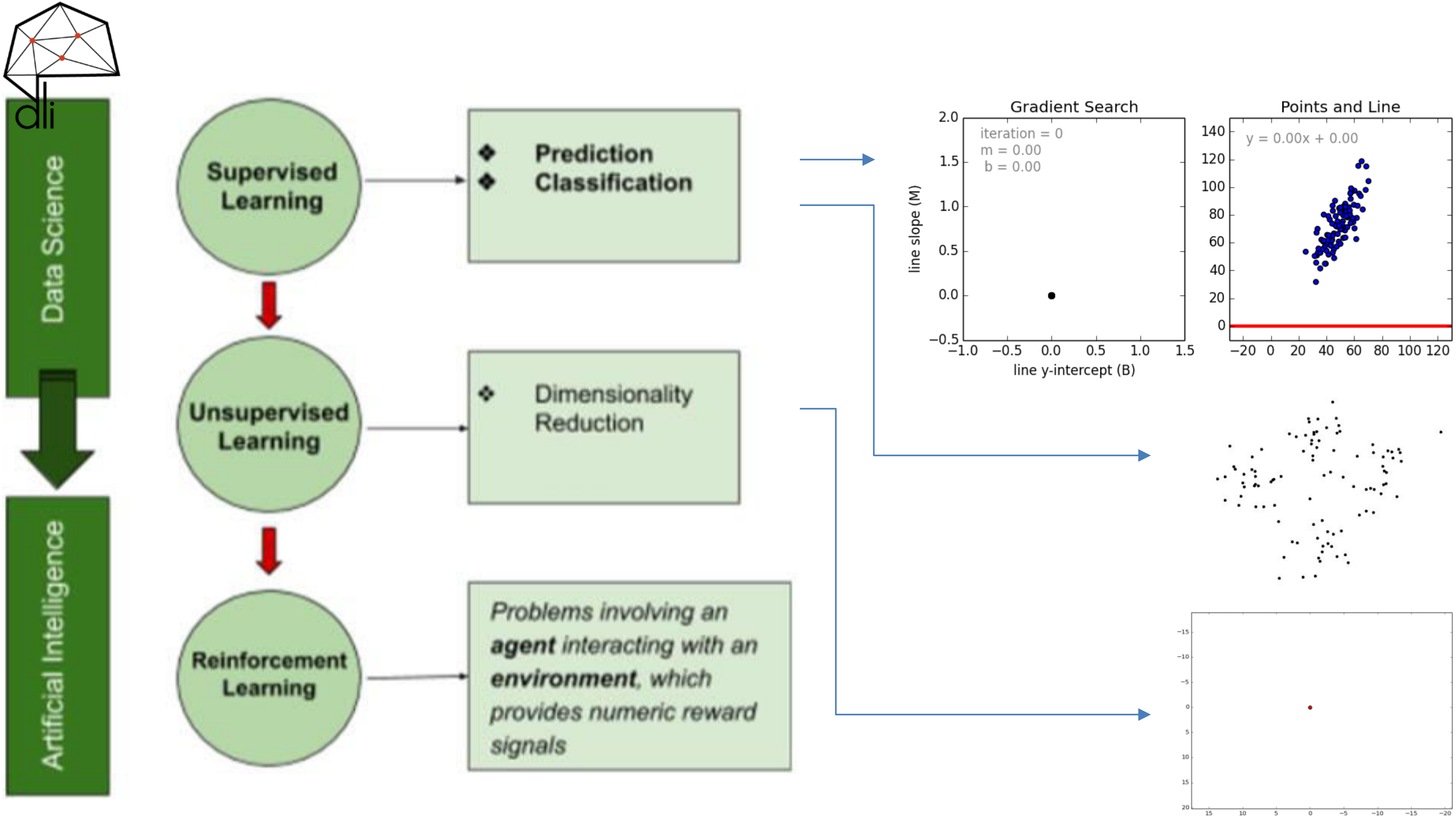
Train AutoEncoder & Predict



```
autoencoder.compile(loss='mean_squared_error', optimizer='adagrad')
AE = autoencoder.fit(X_train, X_train, epochs=10, batch_size=100,verbose=True,
X_tr = encoder.predict(X_train)
X_ts = encoder.predict(X_test)
```

```
Epoch 1/10
29580/29580 [==============================] - 1s 46us/step - loss: 0.8852
Epoch 2/10
29580/29580 [==============================] - 1s 22us/step - loss: 0.6660
Epoch 3/10
29580/29580 [==============================] - 1s 18us/step - loss: 0.6179
Epoch 4/10
29580/29580 [==============================] - 1s 20us/step - loss: 0.5998
Epoch 5/10
29580/29580 [==============================] - 1s 20us/step - loss: 0.5894
Epoch 6/10
29580/29580 [==============================] - 1s 19us/step - loss: 0.5824
Epoch 7/10
29580/29580 [==============================] - 1s 18us/step - loss: 0.5773
Epoch 8/10
29580/29580 [==============================] - 1s 19us/step - loss: 0.5735
Epoch 9/10
29580/29580 [==============================] - 1s 20us/step - loss: 0.5704: 0
s - lo
Epoch 10/10
29580/29580 [==============================] - 1s 20us/step - loss: 0.5679
```

FINE?



Data Science

Artificial Intelligence

Supervised Learning

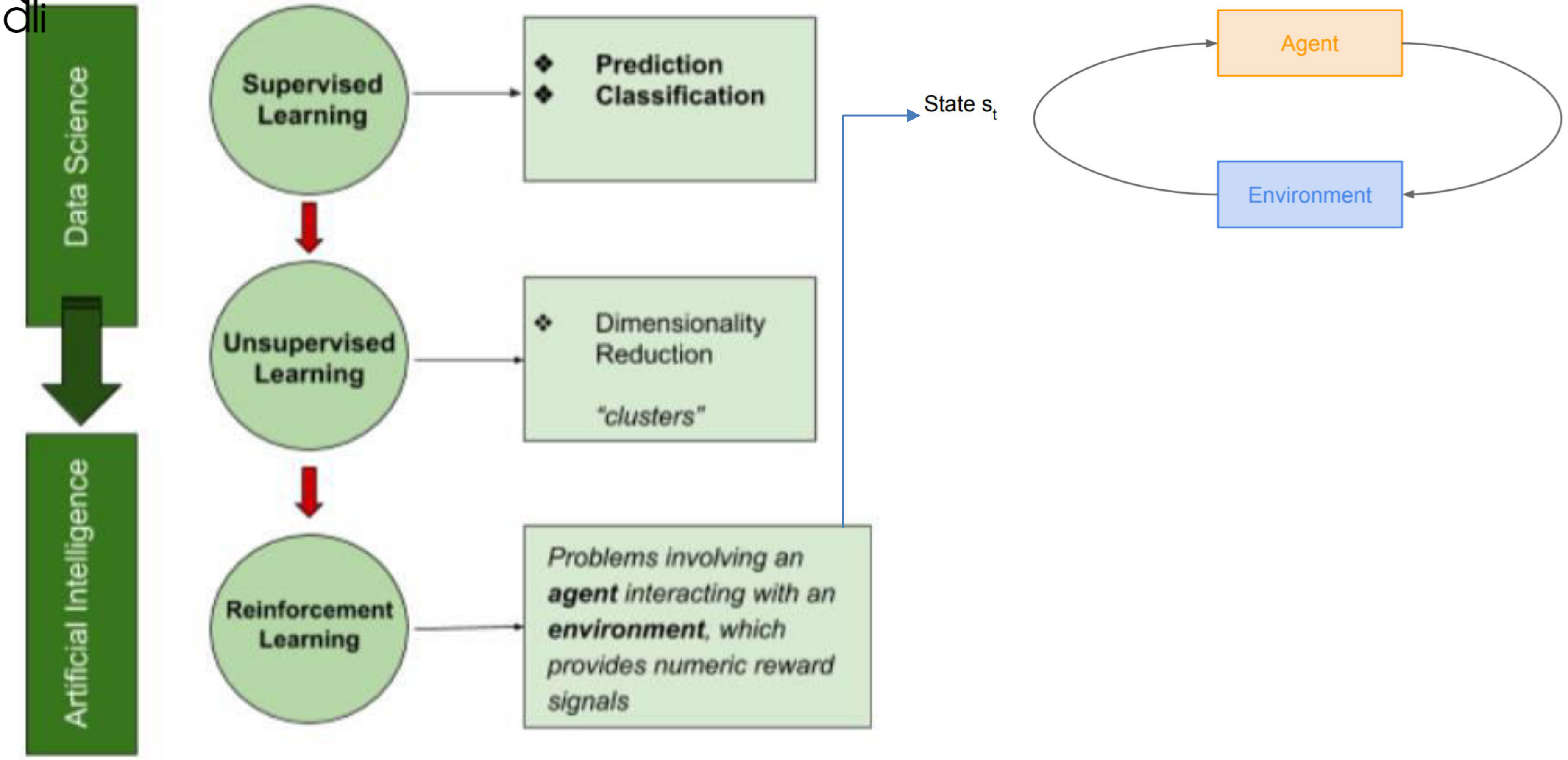
- Prediction
- Classification

Unsupervised Learning

- Dimensionality Reduction

Reinforcement Learning

Problems involving an ***agent*** *interacting with an* ***environment****, which provides numeric reward signals*



Data Science

Artificial Intelligence

Supervised Learning

- **Prediction**
- **Classification**

Unsupervised Learning

- Dimensionality Reduction

"clusters"

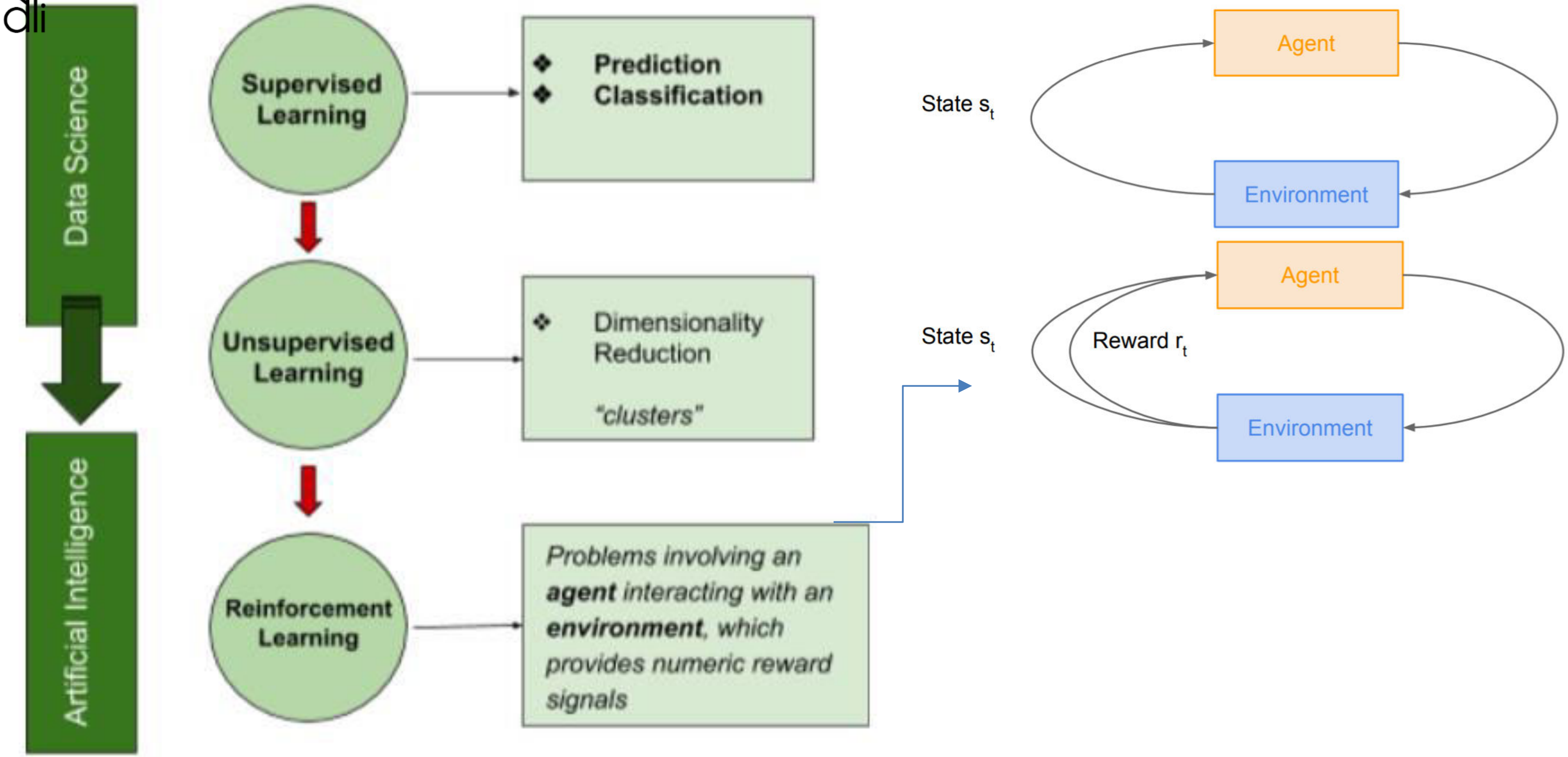
Reinforcement Learning

*Problems involving an **agent** interacting with an **environment**, which provides numeric reward signals*

State s_t

Agent

Environment



Data Science

Artificial Intelligence

Supervised Learning

- Prediction
- Classification

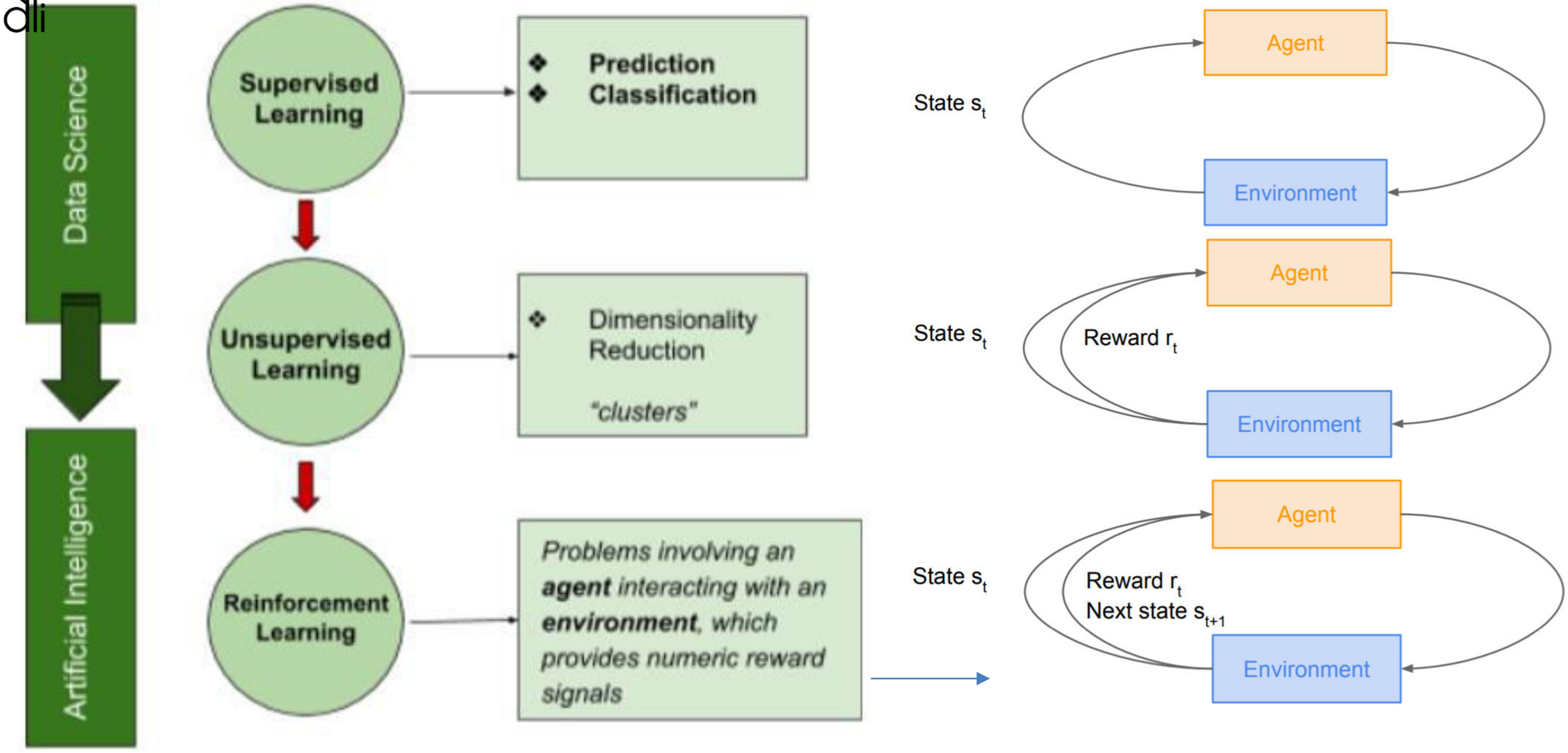
Unsupervised Learning

- Dimensionality Reduction

"clusters"

Reinforcement Learning

Problems involving an ***agent*** *interacting with an* ***environment****, which provides numeric reward signals*



Data Science

Artificial Intelligence

Supervised Learning

- Prediction
- Classification

Unsupervised Learning

- Dimensionality Reduction

"clusters"

Reinforcement Learning

Problems involving an ***agent*** *interacting with an* ***environment****, which provides numeric reward signals*

State s_t

Agent

Environment

State s_t

Reward r_t

Agent

Environment

State s_t

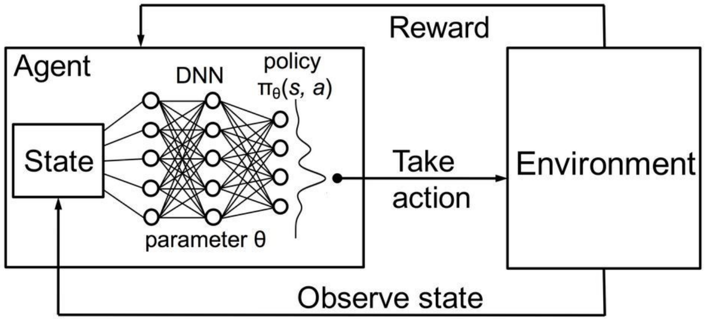
Reward r_t

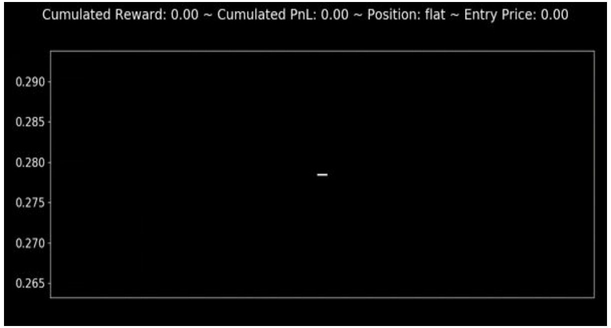
Next state s_{t+1}

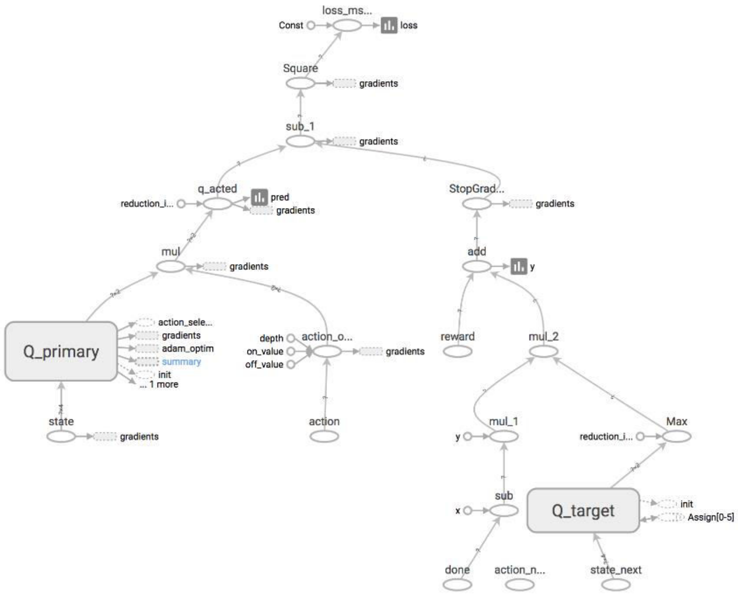
Agent

Environment

Deep Reinforcement Learning




Cumulated Reward: 0.00 ~ Cumulated PnL: 0.00 ~ Position: flat ~ Entry Price: 0.00


loss_ms...
Const
loss
Square
gradients
sub_1
gradients
q_acted
reduction_i...
pred
gradients
StopGrad...
gradients
mul
gradients
add
y
Q_primary
action_sele...
gradients
adam_optim
summary
init
... 1 more
depth
on_value
off_value
action_o...
gradients
reward
mul_2
state
gradients
action
y
mul_1
reduction_i...
Max
x
sub
Q_target
init
Assign[0-5]
done
action_n...
state_next