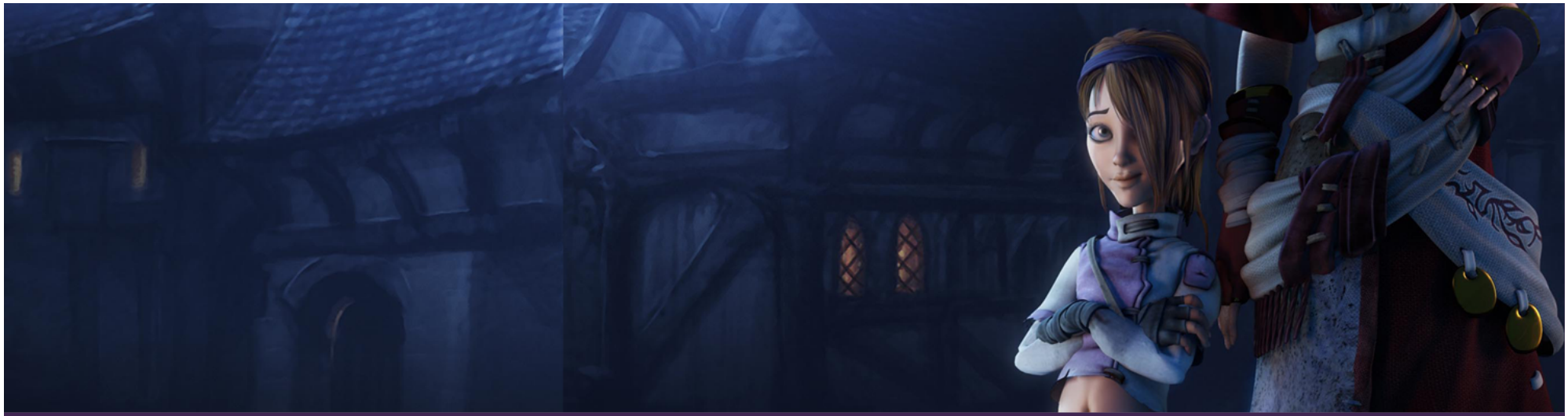


FALMOUTH
UNIVERSITY

Lecture 4: Data Science – Regression & Classification

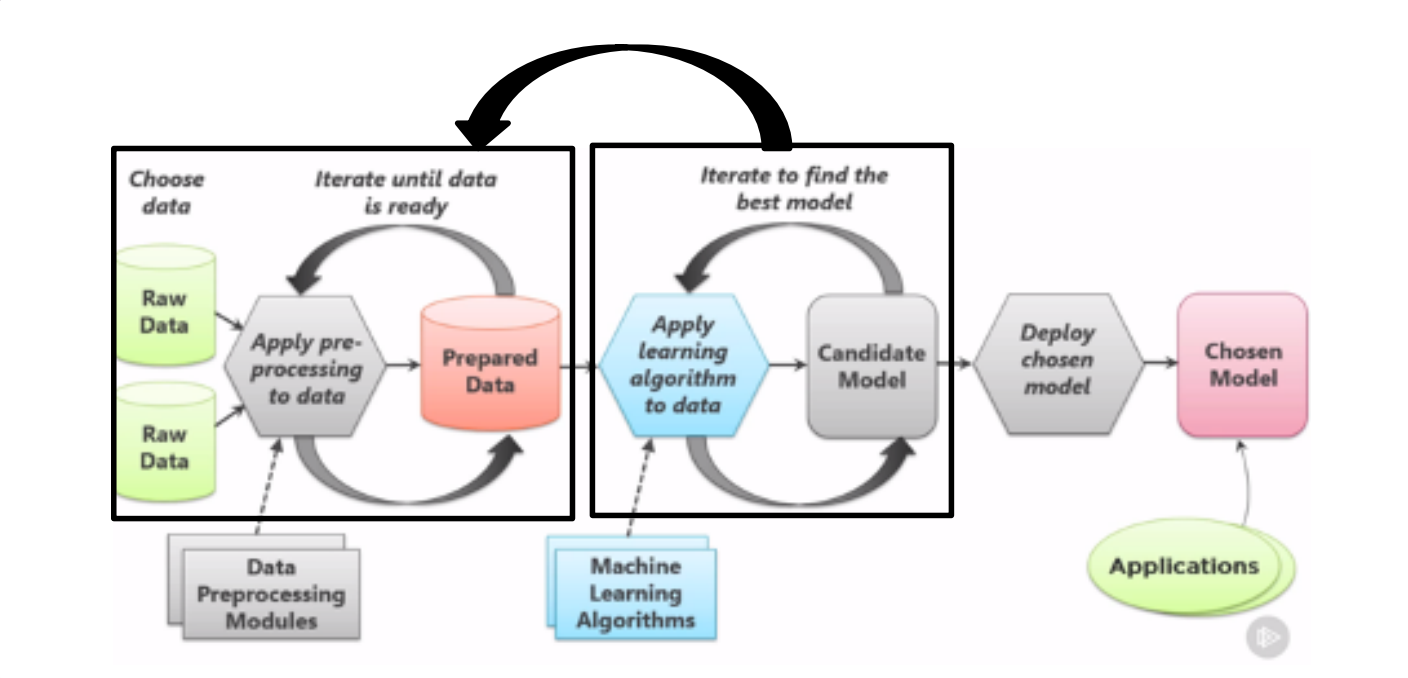
COMP704: Machine Learning
MSc Artificial Intelligence for Games

- Today's session:
  - Wrap up from the regression workshop
  - Classification Learning Algorithm
  - Workshop

- Wrap up from the regression workshop

- Wrap up from the regression workshop
  - Last week, we looked at a couple of datasets (sine() & Boston house prices)
  - What did we learn?
    - ‘performance’ is about understanding level of errors
      - » Under & over-fitting
      - » What is acceptable
      - » Whether errors are universal or localised
    - Fixing issues
      - More data
      - More specific data
      - Hyper parameters -> GAs

- Wrap up from the regression workshop
  - Last week, we looked at a couple of datasets (sine() & Boston house prices)
  - What did we learn?
    - Iteration is key, but what to iterate



- Wrap up from the regression workshop
  - Data complexity
    - On the surface, the house price data looks quite benign
      - There's a lot of it ;)

	year_built	stories	num_bedrooms	full_bathrooms	half_bathrooms	livable_sqft	total_sqft	garage_type	garage_sqft	carport_sqft	has_fireplace	has_pool	has_central_heating	has_central_cooling	city	sale_price
0	1978.00000	1.00000	4.00000	1.00000	1.00000	1689.00000	1859.00000	attached	508.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Hallfort	270897.00000
1	1958.00000	1.00000	3.00000	1.00000	1.00000	1984.00000	2002.00000	attached	462.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Hallfort	302404.00000
2	2002.00000	1.00000	3.00000	2.00000	0.00000	1581.00000	1578.00000	none	0.00000	625.00000	0.00000	0.00000	1.00000	1.00000	Lake Christinaport	2519996.00000
3	2004.00000	1.00000	4.00000	2.00000	0.00000	1829.00000	2277.00000	attached	479.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	197193.00000
4	2006.00000	1.00000	4.00000	2.00000	0.00000	1580.00000	1749.00000	attached	430.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	207897.00000
5	2005.00000	1.00000	3.00000	2.00000	0.00000	1621.00000	1672.00000	attached	430.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	196559.00000
6	1979.00000	1.00000	3.00000	2.00000	1.00000	2285.00000	2365.00000	detached	532.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Morrisport	434697.00000
7	1958.00000	1.00000	5.00000	2.00000	0.00000	1745.00000	1741.00000	none	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	Lake Christinaport	64887.00000
8	1958.00000	1.00000	5.00000	2.00000	0.00000	1747.00000	1745.00000	none	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	Lake Christinaport	143636.00000
9	1961.00000	1.00000	1.00000	1.00000	0.00000	998.00000	1161.00000	none	0.00000	242.00000	0.00000	0.00000	0.00000	0.00000	Lake Christinaport	81896.00000
10	2005.00000	1.00000	4.00000	4.00000	0.00000	2020.00000	2254.00000	attached	502.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	308701.00000
11	2006.00000	1.00000	4.00000	2.00000	1.00000	2517.00000	2679.00000	attached	624.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	333904.00000
12	2003.00000	1.00000	3.00000	1.00000	1.00000	1835.00000	2000.00000	attached	428.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	277197.00000
13	2004.00000	1.00000	4.00000	2.00000	1.00000	2112.00000	2197.00000	attached	397.00000	0.00000	1.00000	1.00000	1.00000	1.00000	Lake Christinaport	351537.00000
14	2007.00000	1.00000	3.00000	3.00000	0.00000	2174.00000	2426.00000	attached	508.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	274681.00000
15	2007.00000	1.00000	3.00000	3.00000	0.00000	2169.00000	2434.00000	attached	509.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	289801.00000
16	1993.00000	1.00000	3.00000	2.00000	0.00000	1646.00000	1648.00000	attached	402.00000	0.00000	1.00000	1.00000	1.00000	1.00000	Lake Christinaport	225538.00000
17	1973.00000	1.00000	2.00000	1.00000	0.00000	835.00000	865.00000	none	0.00000	0.00000	0.00000	0.00000	1.00000	1.00000	Lake Christinaport	56698.00000
18	1989.00000	1.00000	3.00000	2.00000	0.00000	2191.00000	2188.00000	attached	469.00000	0.00000	1.00000	0.00000	1.00000	1.00000	Lake Christinaport	277197.00000

df Format: %.5f

      - Can view this data in PyCharm (select View as … from variable window)

- Wrap up from the regression workshop
  - Data complexity
    - On the surface, the house price data looks quite benign
      - This is a killer statement

```
# Replace categorical data with one-hot encoded data
features_df = pd.get_dummies(df, columns=['garage_type', 'city'])
```

      - Regression (& most other learning algorithms can't work with strings)
      - get_dummies will covert a column of string entries to a set of Boolean columns

	ce	garage_type_attached	garage_type_detached	garage_type_none	city_Amystad	city_Brownport	city_Chadstad	city_Clarkberg	city_Coletown	city_Davidfort	city_Davidtown	ci
0	000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
1	000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
2	0000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
3	00	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
4	000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
5	000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
6	000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
7	00	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.
8	000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.

      - & add *lots* of columns

- Wrap up from the regression workshop
  - Data complexity
    - Regression is working on multiple dimensions of data (60+ in house prices)
    - Hence, the scope to go back to the data model to ‘fix’ issues
      - Can we reduce complexity?

- Wrap up from the regression workshop
  - Data complexity
    - Can we reduce complexity?
      - Does it make sense to have house prices from different cities in one dataset?
        - » Perhaps split them into different models

- » We know that local house prices aren't uniform

- Wrap up from the regression workshop
  - Data complexity
    - Can we reduce complexity?
      - Does it make sense to track the exact price of houses?
      - What about looking at price ranges?
        - » $100,000-$150,000
        - » $150,001-$200,000
        - » etc
      - Putting data into input ‘buckets’ reduces complexity and should lead to better results
      - Likewise for built period, sq.footage and so on

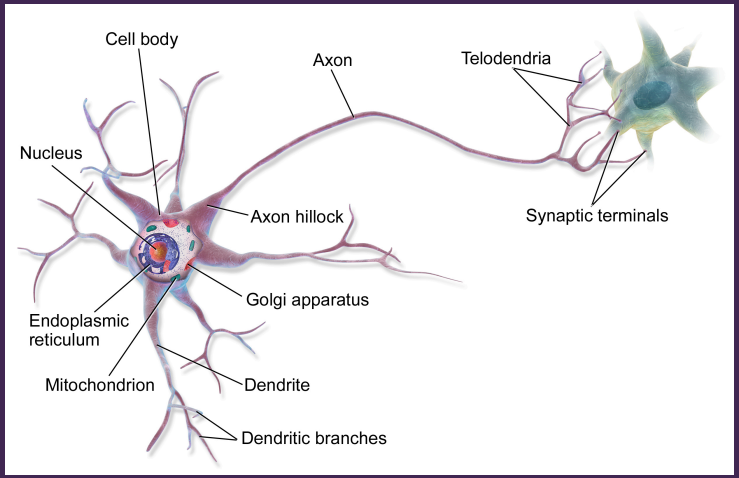
- Wrap up from the regression workshop
  - Data complexity
    - Can we reduce complexity?
      - Does it make sense to track the exact price of houses?
      - What about looking at price ranges?
        - » $100,000-$150,000
        - » $150,001-$200,000
        - » etc
      - Putting data into input ‘buckets’ reduces complexity and should lead to better results
      - Though making the output a bucket will move us away from true regression (lack of continuous output)
        - » Maybe helpful given DS domain considerations

- Wrap up from the regression workshop
  - Data complexity
    - We can see the core activity of machine learning is data processing (or pre-processing)
      - Relies on our understanding of the problem domain (data science)
        - » What data is key to work with
        - » How can data be grouped
        - » What granularity is required
      - This is an iterative process as you can largely guarantee that you wont be right first time (or second time …)

- Classification Learning Algorithm

- Classification Learning Algorithm
  - Classification is used when:
    - We have supervised learning (desired outputs)
    - Discrete (non-continuous) outputs

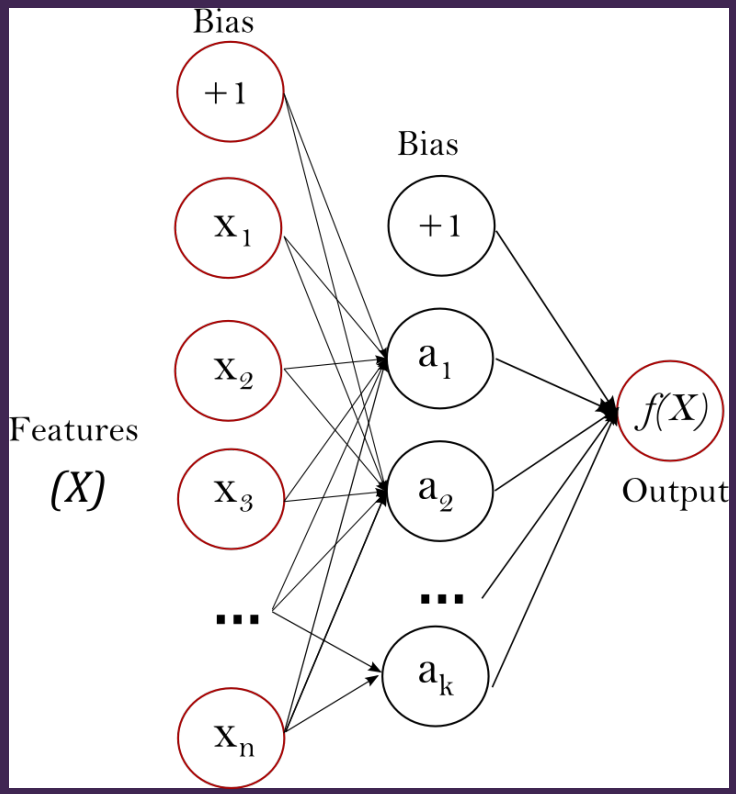
- Classification Learning Algorithm
  - Multilayer perceptron classifiers (MLPClassifier)
  - Perceptrons are a class of neural network
    - Focus of early non-symbolic AI research
    - Geared around loose approximation of human neurons
      - Multiple inputs that can trigger an output



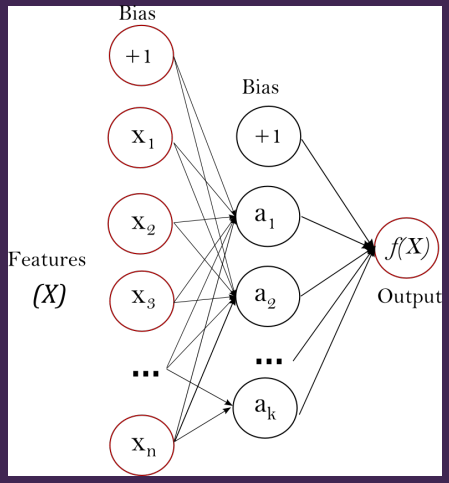
      - From a desire to make AI that is good at solving the kinds of problems that humans are good at solving

- Classification Learning Algorithm
  - Multilayer perceptron classifiers (MLPClassifier)
  - Perceptrons are a class of neural network
    - Focus of early non-symbolic AI research
      - Rosenblatt demonstrated that a single layer perceptron could learn logic gates
        - » The ‘electronic brain’
      - Mintzky demonstrated that they couldn’t be taught XOR
        - » The end of the ‘electronic brain’
    - However, adding hidden layers to the architecture made them far more useful

- Classification Learning Algorithm
  - Multilayer perceptron classifiers (MLPClassifier)
  - Perceptrons are a class of neural network
    - However, adding hidden layers to the architecture made them far more useful

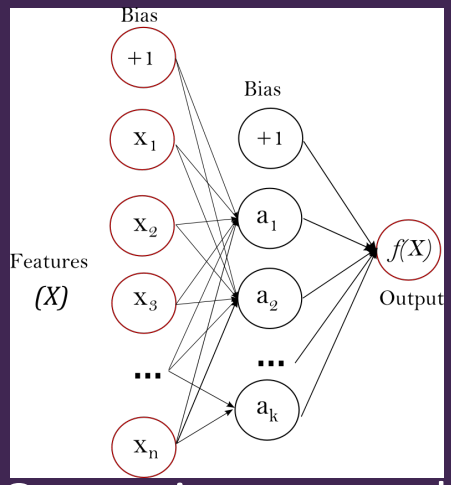


- Classification Learning Algorithm
  - Multilayer perceptron classifiers (MLPClassifier)
    - Process



      - Input (X) are encoded on the left and passed across to the output (Y)
        - » Each node in the next layer will summate its inputs and apply a bias term to determine its output value.

- Classification Learning Algorithm
  - Multilayer perceptron classifiers (MLPClassifier)
    - Learning



      - Output is compared with desired output and differences (error) is propagated back through the network to adjust bias values
        - » Hence ‘back propagation network’ & ‘feedforward network’
      - This is all dealt with by sklearn ☺

- Classification Learning Algorithm
  - Other neural network classifiers
    - Recurrent network
      - As per Mariflow, with ‘memory’
      - Connections are not strictly feed-forward and can form cycles
    - Generative adversarial network
      - A system of 2 neural networks competing with game theory

https://petapixel.com/2020/02/05/youtuber-used-ai-to-upscale-a-classic-1896-short-film-to-4k-and-60fps

- Classification Learning Algorithm
  - Sklearn limitations
    - Neural network training benefits greatly from gpu accelerations
      - All those node calculations lend themselves to vector units
    - Currently, sklearn doesn't support this
      - But other learning frameworks do
        - » Sklearn provides sklearn-style wrappers for them
        - » Or use other frameworks

- Classification Learning Algorithm
  - Let's do some learning
    - 1. Data acquisition (& pre-processing)
      - Sklearn supports two types of BPN / MLP
        - » MLPRegressor
          - Continuous data output
          - Just like the regressor last week
        - » MLPClassifier
          - Discontinuous data output
          - Just like Mariflow
          - We'll need to bucket up the outputs

- Classification Learning Algorithm
  - Let's do some learning
    - 1. Data acquisition (& pre-processing)
      - MLPRegressor
        - » Supplied house price data didn't work very well
          - Lots of inputs
          - Although there's c44,000 data items, there's 50-odd cites
            - c1,000 training items per city
          - Not really any good for training by city
          - Too big to quickly train either (44,000 entries x 60 inputs)

- Classification Learning Algorithm
  - Let's do some learning
    - 1. Data acquisition (& pre-processing)
      - MLPRegressor
        - » To better test, I've built a house model

```
for i in range(0,5000):
    year_built = rand.randint(1940,2020)
    num_bedrooms = rand.randint(1,6)
    total_sqft = num_bedrooms * room_size_factor[rand.randint(0,len(room_size_factor)-1)]
    garage_sqft = garage_size_factor[rand.randint(0,len(garage_size_factor))-1]
    sale_price = ((total_sqft * 200) + (garage_sqft *4)) * ((100 + randrange(-10,10))/100.0)
```

Pros	Cons
Massively scalable	Clear linear model
Clear linear model	
A way to learn how to use pandas	

- Classification Learning Algorithm
  - Let's do some learning
    - 1. Data acquisition (& pre-processing)
      - MLPRegressor
        - » The house model allow me to get clear ideas on the relationships between
          - Data population size
          - Data disparity
          - Iterations of training
          - Over / under-fit issues

- Classification Learning Algorithm
  - Let's do some learning
    - 2.Configure the training algorithm

sklearn.neural_network.MLPRegressor

class sklearn.neural_network.**MLPRegressor**(*hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*) [source]

- Where mulit-layer perceptrons get interesting is that you can define your own hidden layer(s) configuration

```
model = MLPRegressor(max_iter = 2000)
```

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

- Classification Learning Algorithm
  - Let's do some learning
    - 3.Train
    - 4.Evaluate with test data
    - 5.Evaluate performance
    - 6.Iterate or ship?
      - Just like regression learning
      - Results:

```
Training Set Mean Absolute Error: 17077.6051
Test Set Mean Absolute Error: 17448.8123
```

» Not bad, but could do with being smaller (probably)

FALMOUTH UNIVERSITY

- Classification Learning Algorithm
  - Let's do some learning (7,500 population, 50%)

	0	1	2	3	4	5	6	7	8	9	10
0	125326.84386	479994.91122	129865.87730	720744.23941	176692.04236	177241.01008	37713.05526	230110.41570	479502.92647	479940.24624	582576.59874

year_built	num_bedrooms	total_sqft	garage_sqft	sale_price
2023	3	750	0	126000.00000
1952	4	2500	0	523200.00000
1961	3	750	1000	112840.00000
1986	6	3750	500	693120.00000
1996	4	1000	0	158400.00000
2011	2	1000	500	165240.00000
2018	1	250	1000	46640.00000
2009	3	1250	1000	231800.00000
1988	4	2500	0	518400.00000
1956	4	2500	0	432000.00000
2004	5	3000	500	583940.00000
1989	6	3750	0	655200.00000

- Classification Learning Algorithm
  - Let's do some learning
    - 1. Data acquisition (& pre-processing)
      - » MLPClassifier
        - First rule of classifier-club is that you have to group outputs
          - Will fail (crash) if you don't do this

```
for i in range(0,7500):
    year_built = rand.randint(1940,2020)
    num_bedrooms = rand.randint(1,6)
    total_sqft = num_bedrooms * room_size_factor[rand.randint(0,len(room_size_factor)-1)]
    garage_sqft = garage_size_factor[rand.randint(0,len(garage_size_factor))-1]
    sale_price = ((total_sqft * 200) + (garage_sqft *4)) * ((100 + randrange(-10,10))/100.0)

    range_step = 25000

    price_range = int((sale_price + (range_step-1)) /range_step)*range_step

    if not price_range in price_histogram:
        price_histogram[price_range] = 0

    price_histogram[price_range] += 1
    prices.append(price_range)

    sale_price = float(price_range)
```

- Classification Learning Algorithm
  - Let's do some learning
    - 1. Data acquisition (& pre-processing)
      - » MLPClassifier
        - Also recommend to use scaling on input data to make it 'process better'

```
sc = StandardScaler()
features_df = df_in.copy()
del features_df['sale_price']

X = sc.fit_transform(features_df)
```

        - Remember to pickle scale data for app

- Classification Learning Algorithm
  - Let's do some learning
    - 1. Data acquisition (& pre-processing)

	year_built	num_bedrooms	total_sqft	garage_sqft	sale_price
0	2023	3	750	0	125000.00000
0	1952	4	2500	0	475000.00000
0	1961	3	750	1000	125000.00000
0	1986	6	3750	500	775000.00000
0	1996	4	1000	0	150000.00000
0	2011	2	1000	500	150000.00000
0	2018	1	250	1000	50000.00000
0	2009	3	1250	1000	250000.00000
0	1988	4	2500	0	525000.00000
0	1956	4	2500	0	525000.00000
0	2004	5	3000	500	575000.00000
0	1989	6	3750	0	750000.00000
0	2024	1	250	500	50000.00000
0	1995	2	500	500	100000.00000
0	1982	3	1250	0	225000.00000
0	2022	2	500	500	100000.00000
0	1975	2	1000	1000	175000.00000
0	1990	6	3750	500	675000.00000
0	1946	4	1000	1000	175000.00000
0	1973	4	1000	1000	175000.00000
0	1960	4	1750	0	350000.00000
0	1956	1	250	500	50000.00000
0	1975	6	1250	1000	250000.00000
0	1966	4	2500	500	525000.00000
0	2021	3	750	1000	125000.00000
0	1960	1	500	500	100000.00000
0	1985	1	250	0	50000.00000

	0	1	2	3
0	1.62819	-0.29124	-0.80926	-1.47478
1	-1.41578	0.29768	1.08508	-1.47478
2	-1.02992	-0.29124	-0.80926	0.91831
3	0.04190	1.47551	2.43818	-0.27824
4	0.47062	0.29768	-0.53864	-1.47478
5	1.11371	-0.88016	-0.53864	-0.27824
6	1.41382	-1.46908	-1.35051	0.91831
7	1.02797	-0.29124	-0.26802	0.91831
8	0.12764	0.29768	1.08508	-1.47478
9	-1.24429	0.29768	1.08508	-1.47478
10	0.81360	0.88660	1.62632	-0.27824
11	0.17051	1.47551	2.43818	-1.47478
12	1.67106	-1.46908	-1.35051	-0.27824
13	0.42775	-0.88016	-1.07988	-0.27824
14	-0.12960	-0.29124	-0.26802	-1.47478
15	1.58531	-0.88016	-1.07988	-0.27824
16	-0.42970	-0.88016	-0.53864	0.91831
17	0.21339	1.47551	2.43818	-0.27824
18	-1.67301	0.29768	-0.53864	0.91831
19	-0.51545	0.29768	-0.53864	0.91831
20	-1.07280	0.29768	0.27322	-1.47478
21	-1.24429	-1.46908	-1.35051	-0.27824
22	-0.42970	1.47551	-0.26802	0.91831
23	-0.81556	0.29768	1.08508	-0.27824
24	1.54244	-0.29124	-0.80926	0.91831
25	-1.07280	-1.46908	-1.07988	-0.27824
26	-0.00098	-1.46908	-1.35051	-1.47478

- Classification Learning Algorithm
  - Let's do some learning
    - 2.Configure the training algorithm

sklearn.neural_network.MLPClassifier

class sklearn.neural_network.**MLPClassifier**(*hidden_layer_sizes=(100,), activation='relu', solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*) [source]

- Where the MLPClassifier gets interesting is that you can define your own hidden layer(s) configuration

```
model = MLPClassifier(max_iter = 1000)
```

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

- Classification Learning Algorithm
  - Let's do some learning
    - 3.Train
    - 4.Evaluate with test data
    - 5.Evaluate performance
    - 6.Iterate or ship?
      - Just like regression learning
      - Results:

```
Training Set Mean Absolute Error: 13506.6667
Test Set Mean Absolute Error: 16502.2222
```

» Again, not bad, but could do with being smaller (probably)

- Classification Learning Algorithm
  - Let's do some learning (7,500 pop & 75% train)

	0	1	2	3	4	5	6	7	8	9	10	11
0	125000.00000	500000.00000	150000.00000	775000.00000	175000.00000	175000.00000	50000.00000	250000.00000	525000.00000	500000.00000	575000.00000	775000.00000

	year_built	num_bedrooms	total_sqft	garage_sqft	sale_price
0	2023	3	750	0	125000.00000
0	1952	4	2500	0	500000.00000
0	1961	3	750	1000	125000.00000
0	1986	6	3750	500	700000.00000
0	1996	4	1000	0	175000.00000
0	2011	2	1000	500	175000.00000
0	2018	1	250	1000	50000.00000
0	2009	3	1250	1000	250000.00000
0	1988	4	2500	0	450000.00000
0	1956	4	2500	0	500000.00000
0	2004	5	3000	500	650000.00000
0	1989	6	3750	0	725000.00000
0	2024	1	250	500	50000.00000

- Classification Learning Algorithm
  - Let's do some learning (7,500 pop & 75% train)
    - Storing the house prices as a histogram gives us some insight into data occurrence

175000 (4888562224) = {int}	653
325000 (4888562032) = {int}	189
350000 (4662570896) = {int}	321
400000 (4888562000) = {int}	268
125000 (4888562800) = {int}	489
500000 (4888561936) = {int}	215
200000 (4888561680) = {int}	297
675000 (4888562256) = {int}	111
225000 (4888561904) = {int}	375
100000 (4888563056) = {int}	751
425000 (4888563408) = {int}	125
650000 (4888562128) = {int}	114
450000 (4888563344) = {int}	216
50000 (4888563152) = {int}	439
775000 (4888563088) = {int}	87
275000 (4888563600) = {int}	370

600000 (4888562352) = {int}	80
250000 (4888562192) = {int}	662
375000 (4889042064) = {int}	254
525000 (4889041616) = {int}	197
475000 (4889042576) = {int}	212
550000 (4889041072) = {int}	71
150000 (4889042192) = {int}	452
75000 (4889099184) = {int}	56
575000 (4889096240) = {int}	63
300000 (4889098928) = {int}	99
625000 (4889097520) = {int}	88
700000 (4889125328) = {int}	60
750000 (4889127632) = {int}	64
725000 (4889204624) = {int}	83
800000 (4889251056) = {int}	39

- Classification Learning Algorithm
  - Let's do some learning (7,500 pop & 75% train)
    - Storing the house prices as a histogram gives us some insight into data occurrence
      - Ideally, all ranges should be equal
        - » Equal chance of being in training data
          - 125000 -> 489 occurrences
          - 775000 -> 700000, only 87 occurrences
    - Flipping to a higher train/test ratio
      - Likely to result in over-fitting
      - Need more data in the hard to reach buckets

- Classification Learning Algorithm
  - Let's do some learning
    - 7.Ship it
      - Just like regression, but with the scaler

```
# Save the trained model to a file so we can use it in other programs
joblib.dump(model, 'trained_house_bp_model.pkl')
joblib.dump(sc, 'trained_house_bp_scaler.pkl')
```

```
# Load the model we trained previously
predictor = joblib.load('trained_house_bp_model.pkl')
predictor_sc= joblib.load('trained_house_bp_scaler.pkl')
```

```
#scale the input data with the scale factor we used for the training & test data
homes_to_value = predictor_sc.transform(homes_to_value)
predicted_home_values = predictor.predict(homes_to_value)
```

- Classification Learning Algorithm
  - Conclusions
    - Takes some work to make work
      - Data size!
      - MLP framework is far more likely to throw issues at you in code
      - Results often seem bonkers
      - Need to think about how to bucket input & output data for best use
      - Need to analyse data to determine training / learning issues

- Do you have any questions for me?

- Workshop

- Workshop
  - This week, I want to have a look at:
    - Experiment with synthetic data
    - Experiment with original house price data
    - Covert output data type to multi-nodes