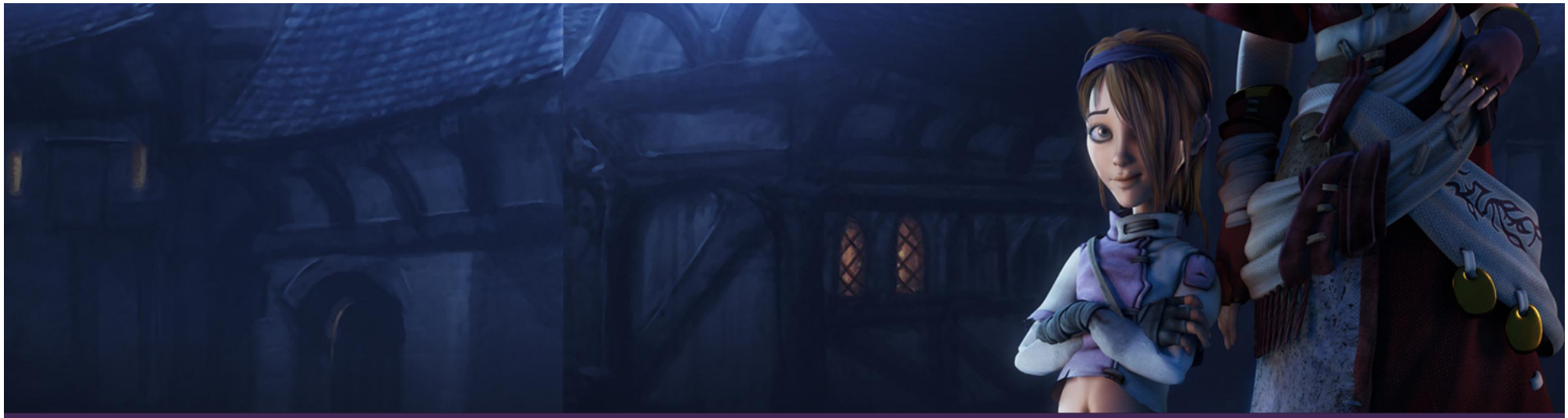


FALMOUTH
UNIVERSITY

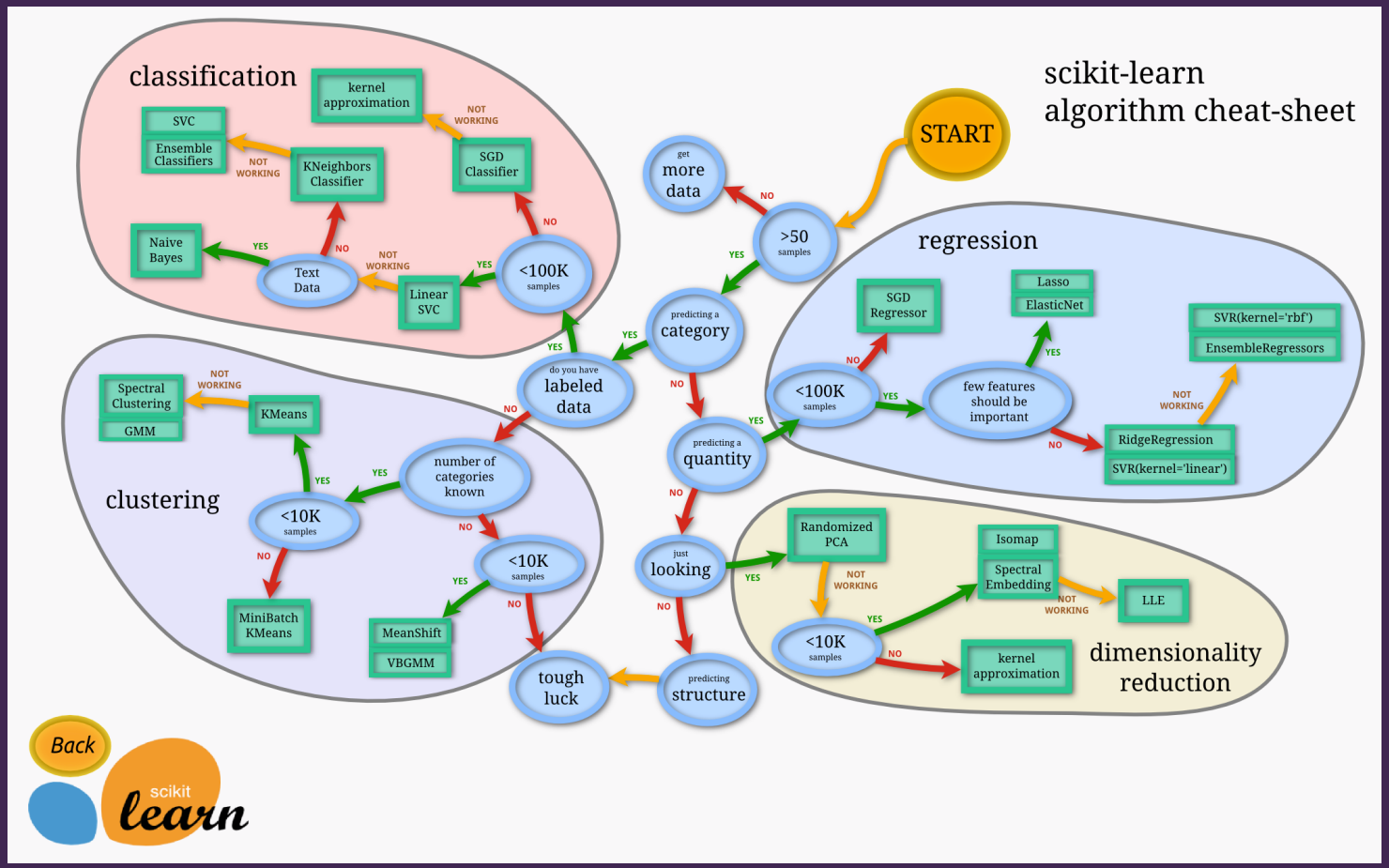
Lecture 3: Data Science - Regression

COMP704: Machine Learning
MSc Artificial Intelligence for Games

- Today's session:
  - Deep dive into the regression learning approach
  - Learning trig through regression
  - Workshop

- Deep dive into the regression learning approach

- Deep dive into the regression learning approach
  - When to use regression?



- Deep dive into the regression learning approach
  - When to use regression?
    - When inputs map to continuous outputs
    - Mariflow
      - Does not have this as inputs map to button presses

- Deep dive into the regression learning approach
  - When to use regression?
    - When inputs map to continuous outputs
    - Crappy breakout
      - Could have this as inputs (the state of the world) can map to where the paddle should be (0… screen_x)
      - If we map inputs to button presses (move_left, move_right, do nothing), then regression is **not** a good fit for this learning

- Deep dive into the regression learning approach
  - When to use regression?
    - When inputs map to continuous outputs
    - We can start to see that machine learning (& data science) is about:
      - Data & learning algorithms
      - Rather than code
    - Need to spend a lot of our time thinking about the relationship between data and learning algorithms in order to solve our problems

- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 1. Get data into a form that can be used for training and testing
      - Typically, use pandas to load data into a dataframe object
      - Split the frame into
        - » X – the inputs
        - » Y – the output
      - Use scikit's train_test_split to create training & testing datasets
        - » X_train & Y_train – training inputs and outputs
        - » Y_test & y_test – testing inputs & outputs
        - » Data is randomised into groups to stop local clustering

- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 2. Choose and configure a training algorithm
      - For each type of training that scikit supports (regression, classification, clustering and dimensionality reduction) there are multiple algorithms
        - » Part of ML is about choosing the most suitable algorithm (by trial and error)
        - » The other part of ML is about configuring parameters (hyper parameters) -> remember what I said about genetic algorithms in week 1
      - Scikit website has lots of information about algorithm use cases

- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 3. Train the model
      - All (most) scikit algorthims perform this through model.fit(x_train, y_train)
      - This may involve having a cup of coffee

- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 4. Evaluate training results with test data
      - All (most) scikit algorthims perform this through model.fit(x_train, y_train)
      - This may involve having a cup of coffee

- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 5. Evaluate performance
      - Use the mean absolute error of training data & test data to give over-fit / under-fit analysis
      - Low error on training & high error on testing
        - » Overfitted algorithm to learn test data by rote rather than learning generalisations
          - Change algorithm / change parameters
      - High error on training
        - » Underfitted algorithm makes random outputs from data
          - Algorithm may not be a good fit for data
          - May not have enough data to train with

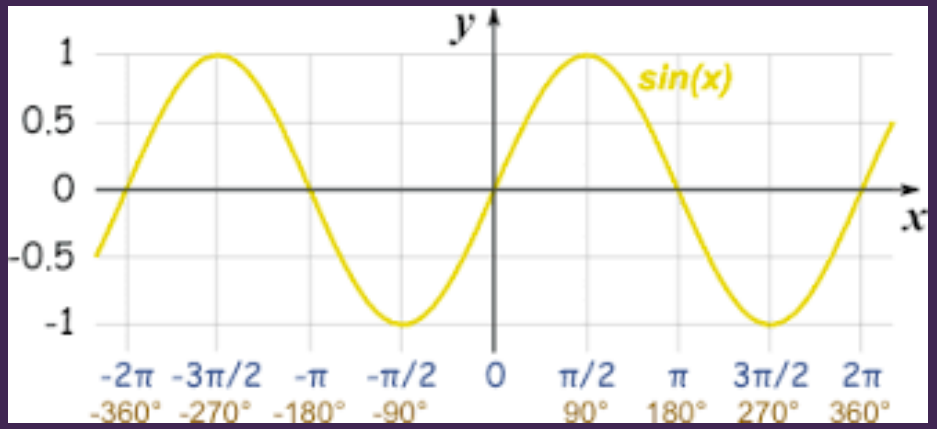
- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 5. Evaluate performance
      - You get to decide what constitutes low & high errors
        - » What may be acceptable to one learning scenario may not be to another
        - » This is where DS & ML meet
          - Error terms likely to be defined as business / domain requirements

- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 6. iterate or ship
      - Either iterate the training process or put the algorithm into ‘production’

- Deep dive into the regression learning approach
  - The scikit-learn approach to regression
    - 7. Ship
      - Save model as a pkl file
      - Write application around pkl data using model.predict
        - » Will take X_frame data (from training) to produce results on novel data

- Learning trig through regression

- Learning trig through regression
  - Regression works when we have a continuous mapping from input(s) to output
    - Sine is a nice & simple example of that we can use to experiment with ML
      - Easy to get lots of data
      - Easy to process



- Learning trig through regression
  - 1. Get data into a form that can be used for training and testing
    - We need to have our data organised as a table of input and output values
    - To generate the sine table data

```
import pandas as pd
# make a sine table and load into pandas dataframe
input_data = []
output_data = []

for angle in range (0,360,1):
    input_data.append(angle)
    output_data.append( math.sin( (angle *math.pi)/180.0))

data = {'angle' : input_data
       ,'value' : output_data}

df = pd.DataFrame (data, columns = ['angle','value'])
```

https://datatofish.com/create-pandas-dataframe/

- Learning trig through regression
  - 1. Get data into a form that can be used for training and testing
    - To get the data into the x & y forms

```
# Create the X and y arrays
x_input = df['angle'].values
x_input = x_input.reshape(-1, 1)
y_output = df['value'].values

# Split the data set in a training set (25%) and a test set (75%)
x_train, x_test, y_train, y_test = train_test_split(x_input, y_output, test_size=0.75, random_state=0)
```

    - Pandas doesn't like 1-D data, so the reshape(-1,1) will put the data into a format it likes

https://stackoverflow.com/questions/35082140/preprocessing-in-scikit-learn-single-sample-depreciation-warning

- Learning trig through regression
  - 2. Choose and configure a training algorithm
    - Let's use linear regression

```
# Fit regression model
model = LinearRegression()
```

      - Typically, each ML algorithm will have attributes that help it learn. These can be set and configured as part of the ML process, though the default values should provide some learning

sklearn.linear_model.LinearRegression

class sklearn.linear_model. **LinearRegression**(*fit_intercept=True, normalize=False, copy_X=True, n_jobs=None*) [source]

https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html?highlight=linearregression

- Learning trig through regression
  - 3. Train the model

```
print('doing training ...')
model.fit(x_train, y_train)
print('done training ...')
```

    - Again, scikit learning algorithms have a fairly common interface taking x_train (inputs) and y_train (outputs)
      - Assuming supervised learning

- Learning trig through regression
  - 4. Evaluate the outcomes

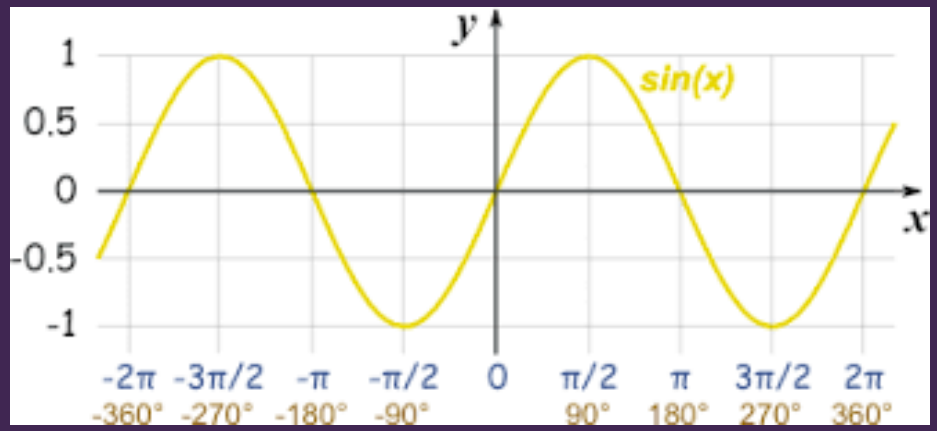
```
# Find the error rate on the training set
mse = mean_absolute_error(y_train, model.predict(x_train))
print("Training Set Mean Absolute Error: %.4f" % mse)

# Find the error rate on the test set
mse = mean_absolute_error(y_test, model.predict(x_test))
print("Test Set Mean Absolute Error: %.4f" % mse)
```

- Learning trig through regression
  - 5. Evaluate performance

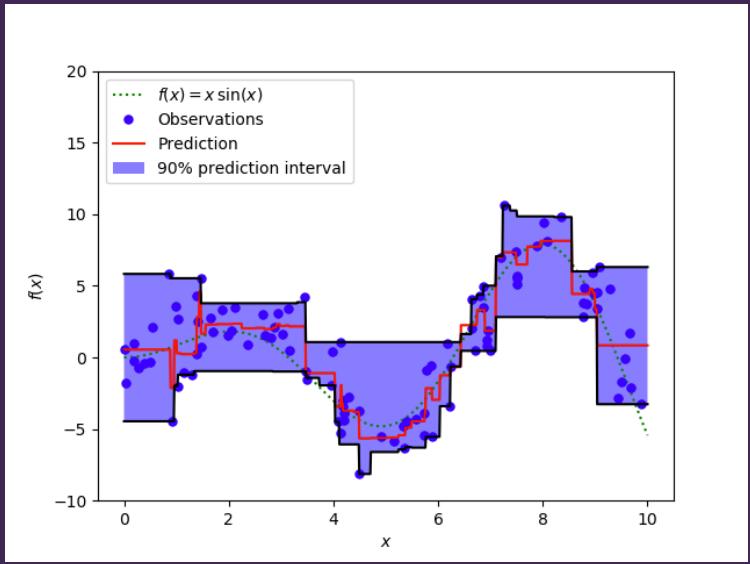
```
Training Set Mean Absolute Error: 0.3418
Test Set Mean Absolute Error: 0.4042
```

    - What do these numbers mean?
      - Sine() ranges from -1 to +1
      - An error of 0.35 for the training data -> underfitting
        - » Algorithm isn't leaning
      - An error of 0.4 for the test data
        - » Algorithm is junk

- Learning trig through regression
  - 6. iterate or ship
    - Iterate
      - Why is training so bad?
        - » Can't really do linear regression on non-linear data
        - » Line of best fit (least errors) is y = 0



- Learning trig through regression
  - 6. iterate or ship
    - Iterate
      - Change algorithm to Gradient Boosting Regressor
        - » Made for more complex & non-linear data
        - » Data is stored in buckets of linear regression



https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

- Learning trig through regression
  - 6. iterate or ship
    - Iterate
      - Create a new model

```
model = ensemble.GradientBoostingRegressor()
```

      - Train it and get the results

```
Training Set Mean Absolute Error: 0.0012
Test Set Mean Absolute Error: 0.0293
```

        - » Training error very low
        - » Test error low, but higher than training
          - Might be worth tweaking the algorithm parameters

- Learning trig through regression
  - 6. iterate or ship
    - Iterate
      - Might be worth tweaking the algorithm parameters

3.2.4.3.6. sklearn.ensemble.GradientBoostingRegressor

class sklearn.ensemble.**GradientBoostingRegressor**(*loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, presort='deprecated', validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0*) [source]

» This is where GAs can become very useful ;)

- Learning trig through regression
  - 7. Ship
    - Embed the model into an application
      - Save model data

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

      - Use model in app

```
new_model = joblib.load('trained_model.pkl')

for angle in range(0, 360):
    test_data = [[angle]]
    model_value = new_model.predict(test_data)
    actual_value = math.sin( (angle *math.pi)/180.0)

    msg = str(angle)
    msg += ": "
    msg += str(round(model_value[0], 3))
    msg += ' ['
    msg += str(round(actual_value, 3) )
    msg += '] err:'
    msg += str(round(math.fabs(model_value[0] - actual_value), 3))
    print(msg)
```

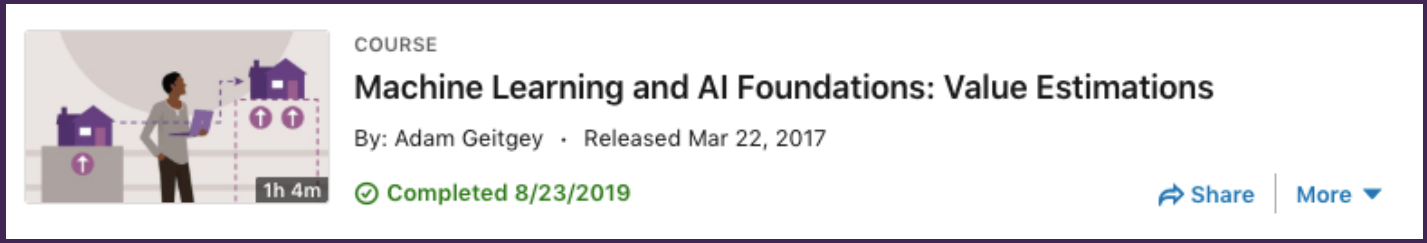
Note generally gnarliness in using model, input data needs to be an array of arrays

- Learning trig through regression
  - 7. Ship
    - Is the algorithm any good in the real world?
      - Test it and find out

```
0: 0.158 [0.0] err:0.158
30: 0.515 [0.5] err:0.015
60: 0.838 [0.866] err:0.028
90: 0.998 [1.0] err:0.002
120: 0.875 [0.866] err:0.009
150: 0.481 [0.5] err:0.019
180: 0.0 [0.0] err:0.0
210: -0.515 [-0.5] err:0.015
240: -0.883 [-0.866] err:0.017
270: -0.996 [-1.0] err:0.004
300: -0.927 [-0.866] err:0.061
330: -0.5 [-0.5] err:0.0
360: -0.036 [-0.0] err:0.036
```

FALMOUTH
UNIVERSITY

- Learning through LinkedIn Learning
  - A lot of the process for this weeks' lecture comes from
    - Machine Learning and AI Foundations: Value Estimations (Adam Geitgey)



      - Uses house prices as an example
      - Well worth having a watch (1hr)
      - You are all subscribed to LinkedIn Learning through your falmouth.ac.uk accounts

https://www.linkedin.com/learning/machine-learning-and-ai-foundations-value-estimations/overfitting-and-underfitting

- Workshop

- Workshop
  - This week, I want to have a look at:
    - Technical design to help you build applications
      - Using UML & Booch to better understand class architecture and game states
    - Using HTTP to manage data
    - Breakout as a regression-based learning model
      - Managing data

- Do you have any questions for me