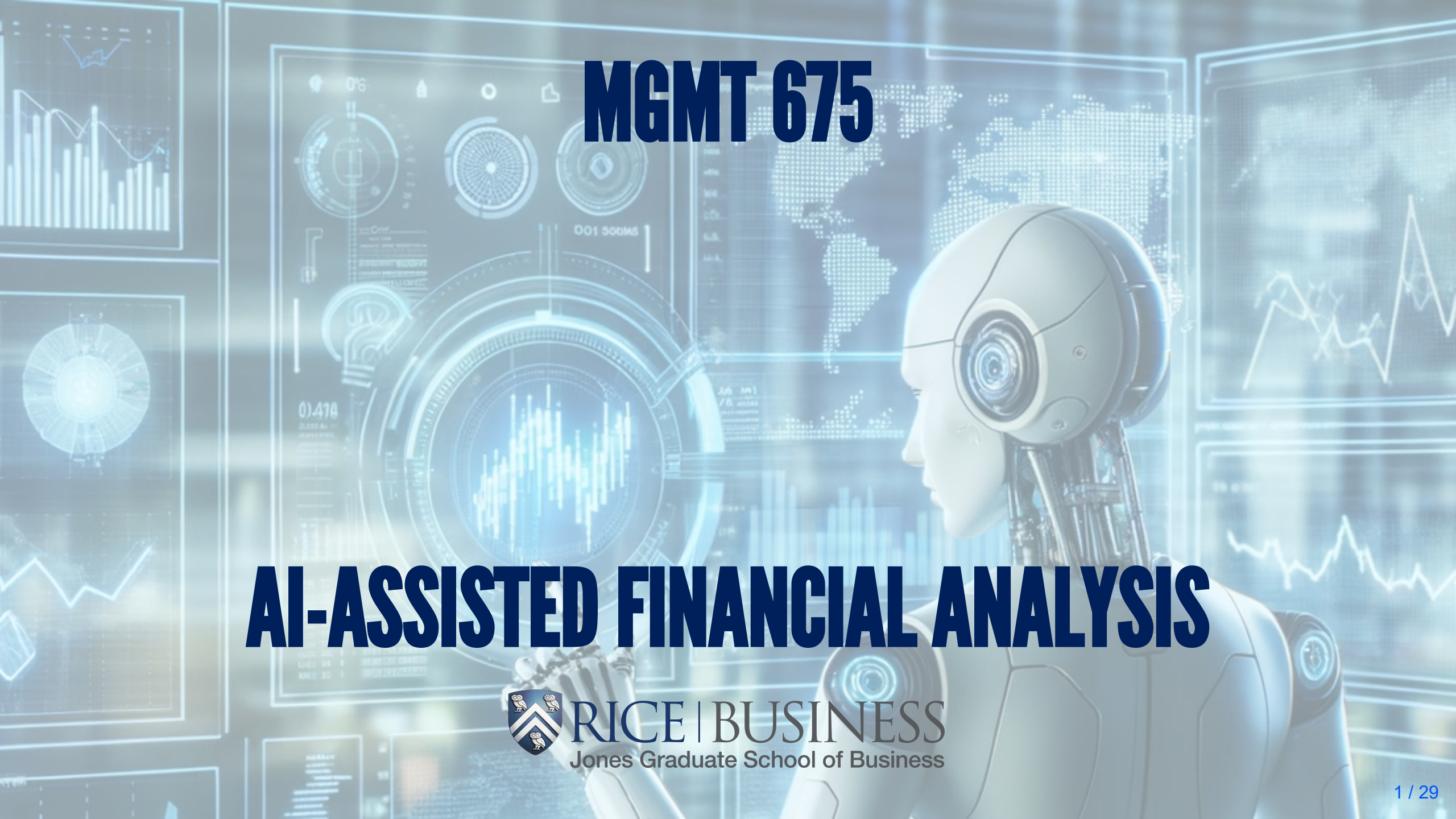
MGMT 675

AI-ASSISTED FINANCIAL ANALYSIS



RICE | BUSINESS
Jones Graduate School of Business

TREES, FORESTS, AND NETS

OUTLINE

- Decision tree
- Random forests
- Gradient boosting
- Neural networks

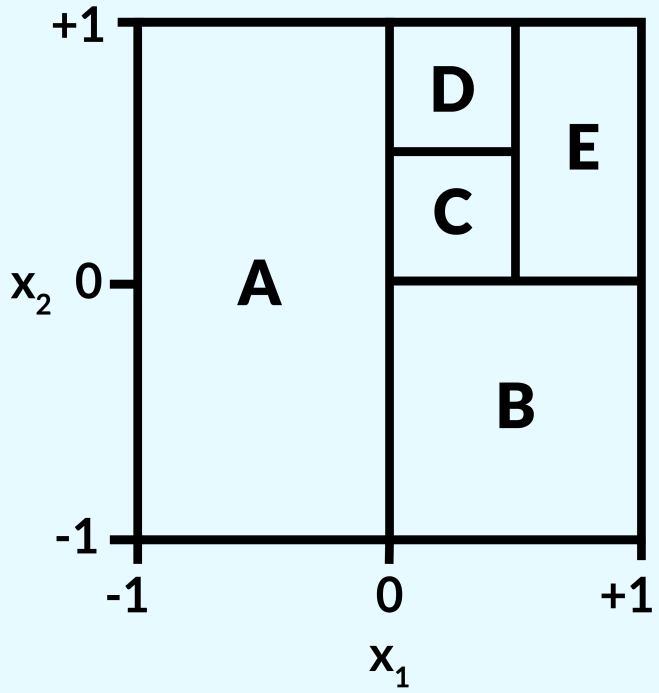
CONCEPTS FROM LAST CLASS

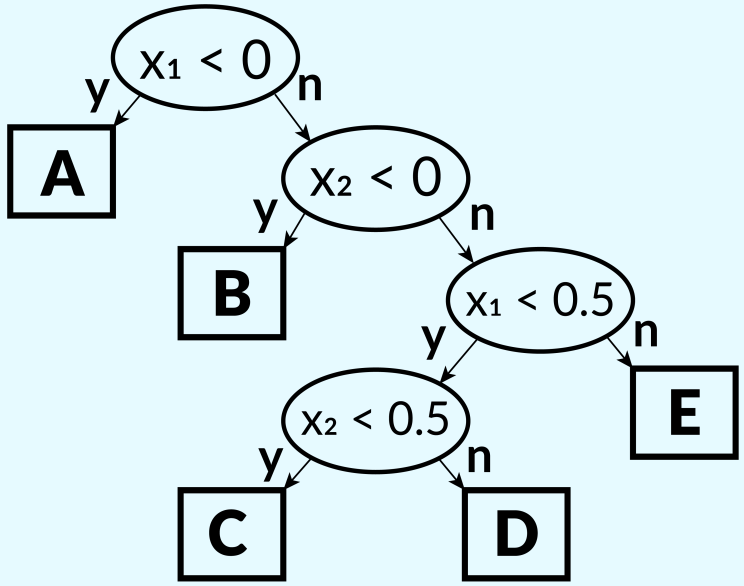
- Train and test
- R-squared (score)
- Underfitting and overfitting
- Hyperparameters
- Cross validation
- Scaling and pipelines

HOW DECISION TREES WORK

- Start with the estimate $\hat{y} = \bar{y}$ for all observations.
- Split into two subsets based on one variable and one threshold. All observations below the threshold go into one group. All above go into another.
- Prediction for each group is the group mean of the target variable. Calculate MSE over both groups.
- Choice of variable and threshold on which to split was based on minimizing MSE.
- Split each subset into further subsets and continue.

EXAMPLE OF DECISION TREE SPLITTING





ANOTHER EXAMPLE

- Ask Julius to read mldata.xlsx.
- Ask Julius to fit a decision tree regressor with max_depth=2 to predict “continuous” from x1 through x100.
- Ask Julius to plot the tree.
- Ask Julius to set x to be an array of 100 standard normals. Ask Julius to show x[:10] and ask Julius what the tree predicts for x.
- Ask Julius to use max_depth=3 and plot the tree.

RANDOM FOREST

FORESTS

- A forest is multiple trees.
- For any observation - old or new - each tree makes a prediction.
- Average the predictions to get the final prediction.

GENERATING RANDOM FORESTS

- A random forest is created by generating random datasets and fitting a tree to each.
- A random dataset is generated by randomly drawing rows from the original dataset.
- Default in scikit-learn is to draw with replacement as many rows as in the original and then drop duplicates.

EXAMPLE

- Ask Julius to fit random forest regression to predict “continuous” from x1 through x100 with n_estimators=2 and max_depth=2.
- Ask Julius to plot both trees.
- Ask Julius what the random forest predicts for x.

A MORE REALISTIC EXAMPLE

- Ask Julius to fit a random forest regression to predict “continuous” from x1 through x100 (let Julius choose n_estimators and max_depth - will probably use defaults).
- Ask Julius what the score is on the training and test data.

IMPORTANT HYPERPARAMETERS

- How much splitting to do
- Examples:
  - max_depth = 3 means split 3 times (2^3 leaves)
  - min_samples_split = = 50 means don't split groups smaller than 50
  - min_samples_leaf = 50 means don't create groups smaller than 50

OTHER IMPORTANT HYPERPARAMETERS

- How to split
  - criterion (squared error, absolute error, …). Absolute error is less influenced by outlier values for the target variable.
  - max_features: Randomly choose n features at each split and split on one of them. Small max_features generates more variation in the trees.
- Number of trees (n_estimators)

EXAMPLE

- Ask Julius to use GridSearchCV to find the best max_depth in [2, 4, 6, 8, 10]
- Ask what the scores are on the training and test data.
- Ask Julius to plot the test data predictions against x1 in the test data.
- Ask Julius to tell you the feature importances.

ANOTHER EXAMPLE

- Ask Julius to get the Boston house price data from sklearn.
- Build a random forest model to predict MEDV using the other variables.
  - GridSearchCV for max_depth
  - Get score on test data
  - Get feature importances

BOOSTING

HOW GRADIENT BOOSTING WORKS

- Fit a decision tree.
- Look at its errors. Fit a new decision tree to predict the errors.
- New prediction is original plus a fraction of the prediction of original's error (fraction = learning rate).
- Look at the errors of the new predictions. Fit a new decision tree to predict these errors.
- Continue …

KEY HYPERPARAMETERS

- Same as random forest
- Plus learning rate

EXTREME GRADIENT BOOSTING (XGBOOST)

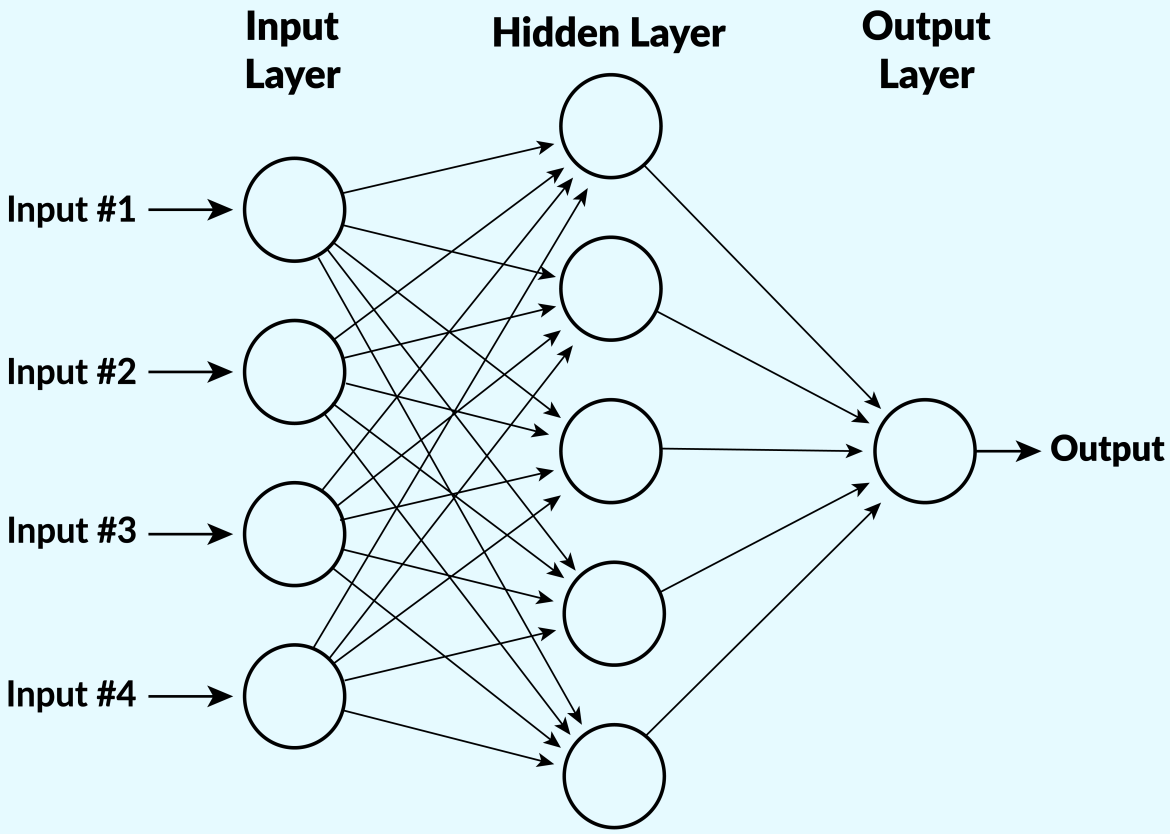
- Ask Julius to explain xgboost
- Ask Julius to fit xgboost to predict “continuous” from x1 through x100 in mldata.xlsx.
- Ask Julius to use GridSearchCV to find the best max_depth and learning rate.

Ask Julius

- what the score is on the test data
- what the feature importances are
- to plot the actual and predicted target values in the test data against x1 in the test data.

NEURAL NETWORKS

EXAMPLE OF MULTI-LAYER PERCEPTRON



RECTIFIED LINEAR UNITS

- The usual function for the neurons (except in the last layer) is

$$y = \max(0, b + w_1 x_1 + \cdots + w_n x_n)$$

- Parameters b (called bias) and $w_1, \ldots w_n$ (called weights) are different for different neurons.
- This function is called a rectified linear unit (ReLU).

ANALOGY TO NEURONS FIRING

- If $w_i > 0$ then $y > 0$ only when x_i are large enough.
- A neuron fires when it is sufficiently stimulated by signals from other neurons (in prior layer).

OUTPUT FUNCTION

- The output doesn’t have a truncation, so it can be negative.
- For regression problems, it is linear:

$$z = b + w_1 y_1 + \cdots + w_n y_n$$

KEY HYPERPARAMETERS

- Number of hidden layers
- Number of neurons in each layer
- Activation function
- Also, choice of optimizer can matter

EXAMPLE

- Ask Julius to fit a multi-layer perceptron to predict “continuous” from x1 through x100 in mldata.xlsx.
- Try different hidden layer sizes. For example (64, 32) means two hidden layers with 64 neurons in the first and 32 in the second.
- You can use GridSearchCV to search over different hidden layer sizes - e.g. (8,), (4, 4, 4), etc.