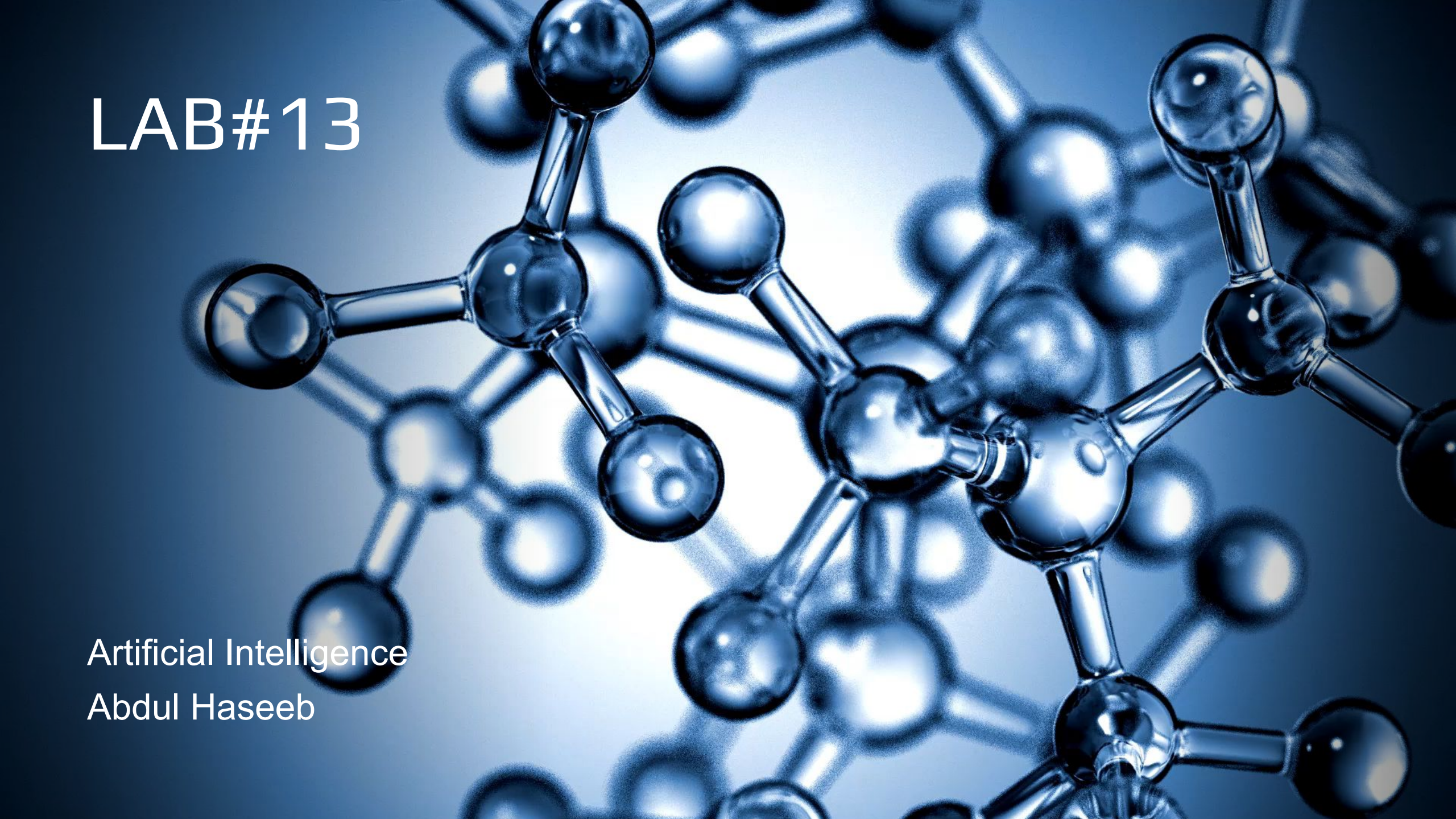
LAB#13



Artificial Intelligence

Abdul Haseeb

Lab Objectives

Implementation of Neural Network using Keras Sequential API

Fine-Tuning Keras Neural Network

Keras Sequential API

- **Model Building Steps:**
  - Specify Architecture:
    - Number of Neurons in Input Layer
    - Number of Neurons in Output Layer
    - Number of Hidden Layers, and Number of Neurons in each Hidden Layer
    - type of Activation functions used
  - Compile the Model:
    - Specify Optimizer to control learning rate
    - Specify Loss function: Depends on Type of problem
  - Fit the Model:
    - Backpropagation with Gradient Descent to update the weights
  - Perform Predictions

Code to Build a Keras Model

```
n_cols = predictors.shape[1]
model = Sequential()
model.add(Dense(100, activation='relu', input_shape=(n_cols,)))
model.add(Dense(100, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(predictors, target)
```

Regression using Neural Networks via Keras Sequential API

- Import hourly_wages.csv, and print first five rows
- Separate the target and features
- Get the Number of features
- Build a Keras Model with following Architecture:
  - Input shape must be equal to number of features
  - There must be three hidden layers, first with 128 neurons, second with 64 and third with 32 neurons, all with relu activation
  - Add one output layer with one neuron
- Compile the Model with adam optimizer and mean_squared_error as the loss (As you are working with regression)
- Fit the model

Classification using Neural Networks via Keras Sequential API

- Import titanic_all_numeric.csv, and print first five rows
- Separate the target and features
- Get the Number of features
- Build a Keras Model with following Architecture:
  - Input shape must be equal to number of features
  - There must be five hidden layers, first with 128 neurons, second with 64, third with 32 neurons, fourth with 16 neurons and fifth with 8 neurons, all with relu activation
  - Add one output layer with two neurons and softmax activation
- Compile the Model with adam optimizer and categorical_crossentropy as the loss (As you are working with Classification), set metric argument to accuracy
- Fit the model