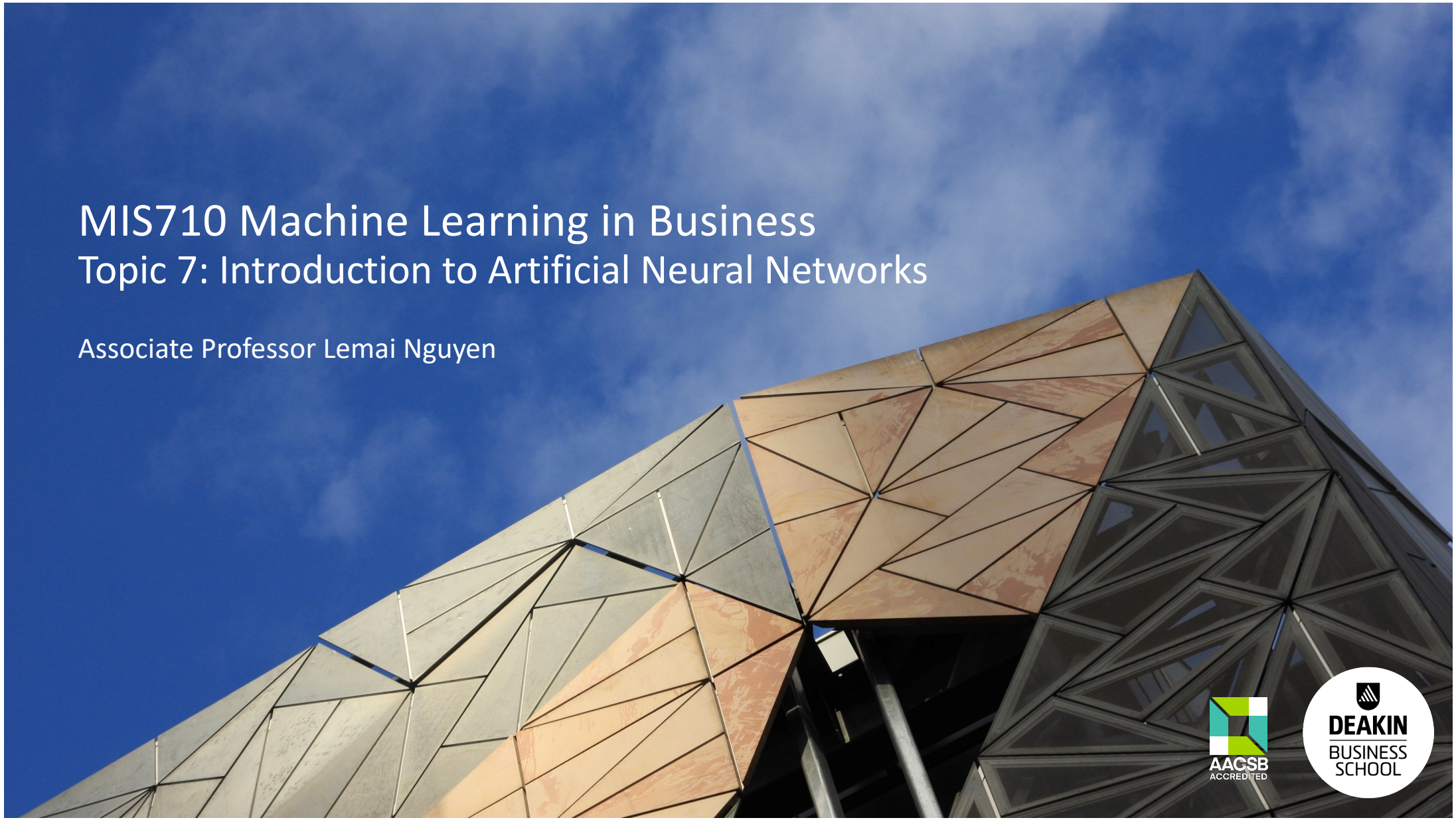


MIS710 Machine Learning in Business

Topic 7: Introduction to Artificial Neural Networks

Associate Professor Lemai Nguyen

AACSB ACCREDITED

DEAKIN BUSINESS SCHOOL

Learning Analytics for Primary Schools

A2

Case Study

A2 Tasks

A2 Deliveries

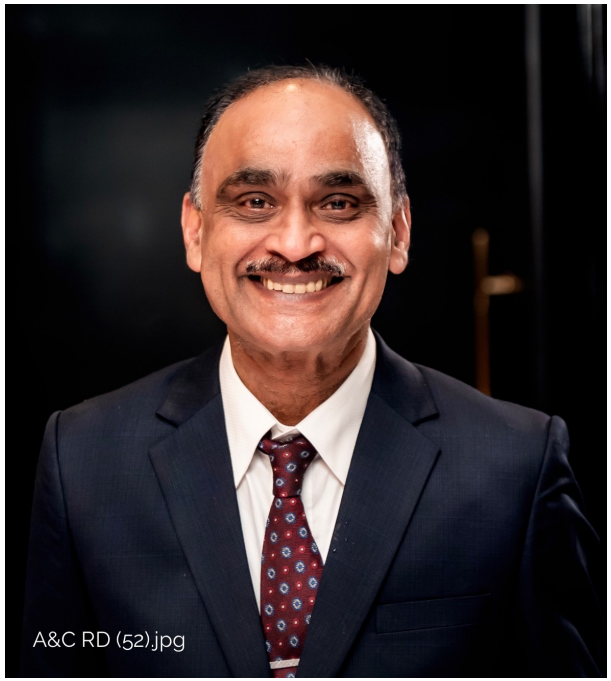
- **Six** EDA questions
- **Two** supervised ML models and evaluation comparison
- **One** clustering model and cluster profiling

- **Two** reports
  - Dr. Alok Sinha, Data2Intel Director of **Data and Insights** and team
  - Sally Tran, Data2Intel Director of Education and Engagement - **business** audience

- **Two** Python files
  - ipynb and PDF version of the Python notebook

Dr. Raju Varanasi

Professor of Practice



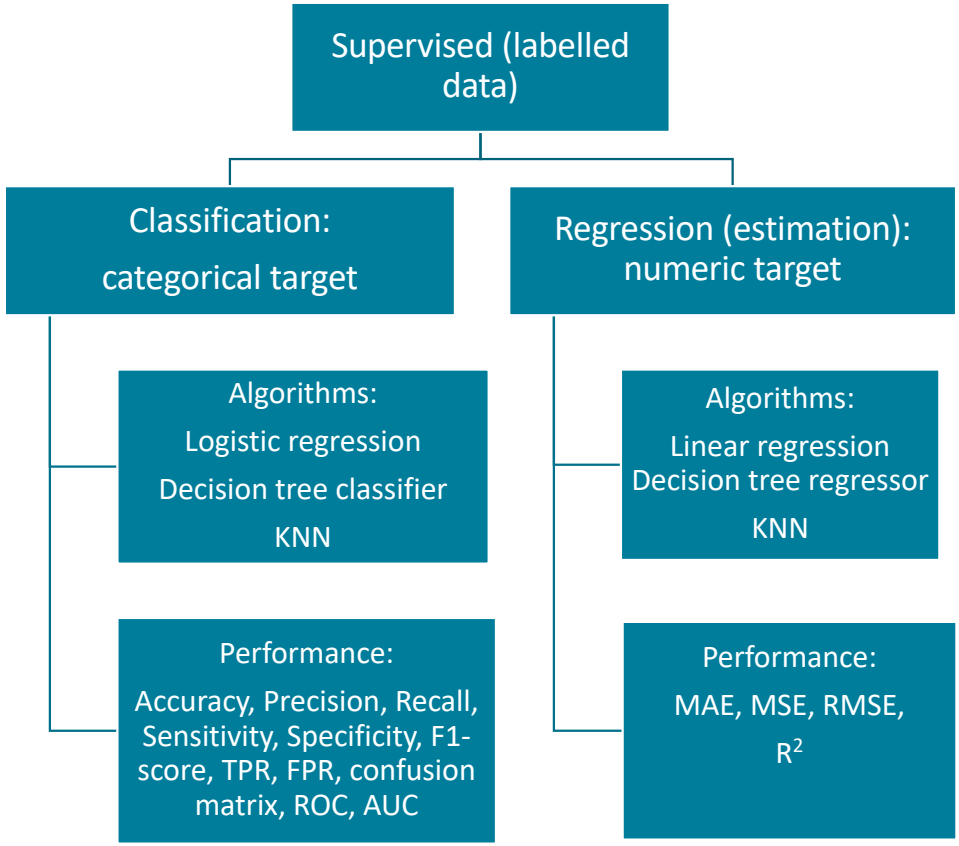
Dr. Raju Varanasi commenced as Professor of Practice in the Department of Information systems and Business Analytics at the Deakin Business School. Dr. Varanasi brings a distinguished blend of educational excellence and industry expertise with a career that spans over three decades in Australia.

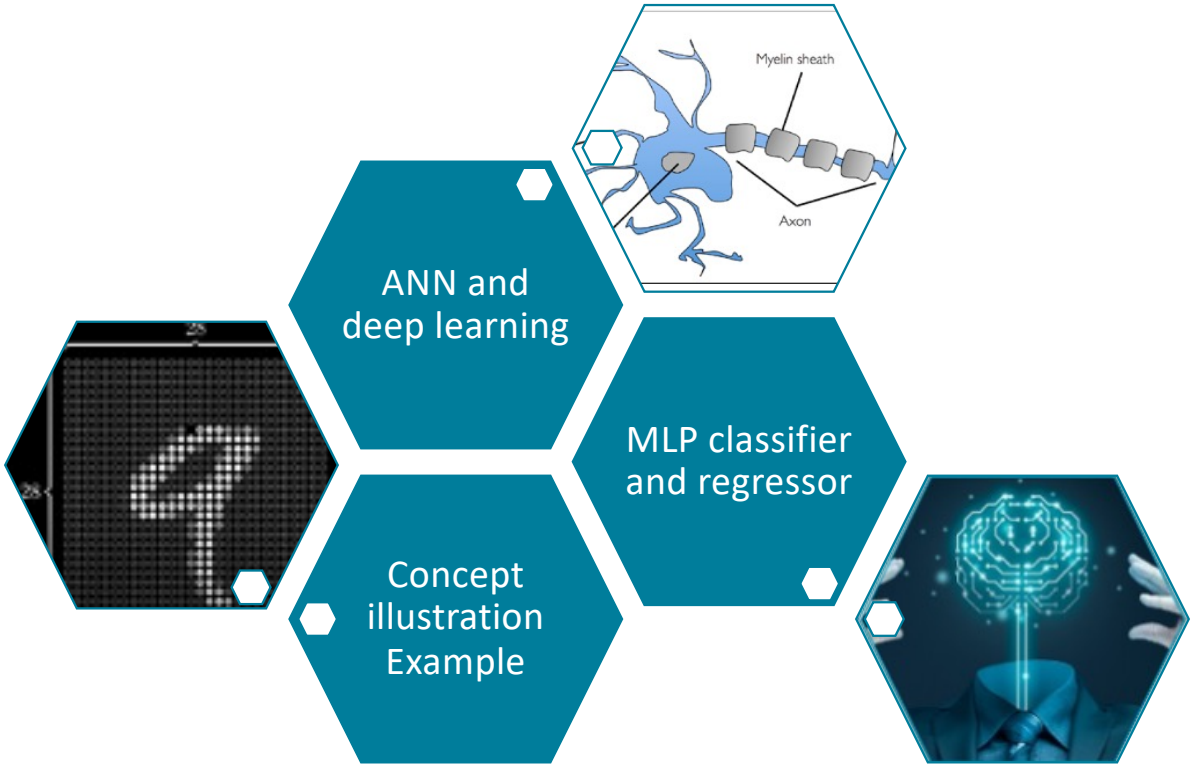
Dr. Varanasi has held pivotal roles in TAFE NSW, public and Catholic schools in leveraging digital technologies and data analytics for transformational impact. Dr. Varanasi is a Chemical Engineer an MBA from IIT, New Delhi and IIM Bangalore- world class universities in India. An Australian Fulbright Scholar, Dr. Varanasi attained his PhD from University of Newcastle with his thesis on transforming school systems. His visionary approaches to education have been recognized with awards such as the Top 50 Australian CIO Award (twice) and the Gartner Innovation in Education Award.

He is also a published author, contributing to the literature on educational analytics and school progress frameworks. After roles as a General Manager, Director, COO and CIO in the last 20 years, Dr. Varanasi is consulting for Google's partners across Asia Pacific advising on data, analytics & AI strategies. As a Professor of Practice, Dr. Varanasi is committed to forging stronger partnerships between academia and industry, preparing students to navigate and excel in the dynamic field of Analytics & AI to equip them with the skills necessary to lead enterprises in a data-driven future.

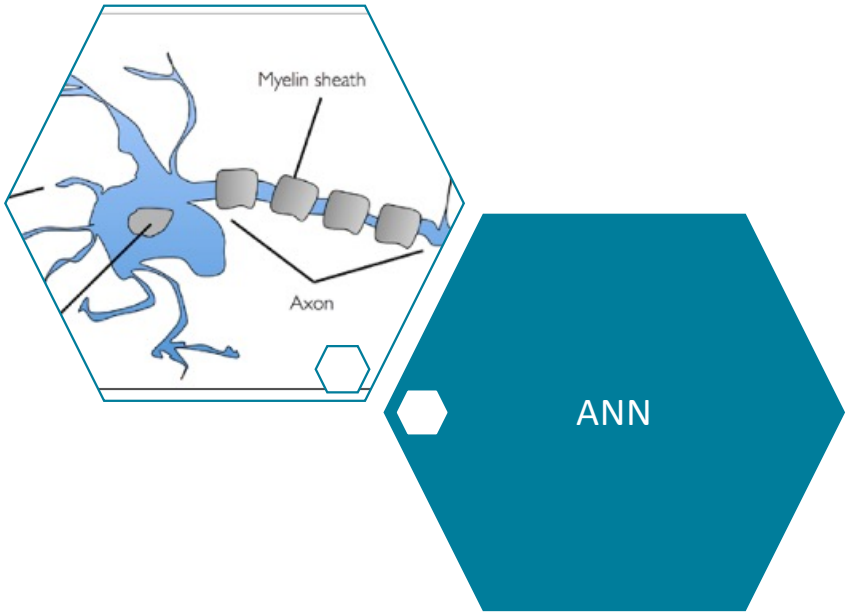
Dr. Raju Varanasi | LinkedIn

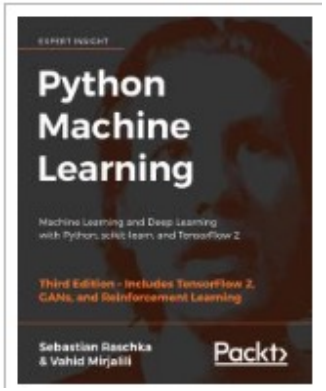
Recap


Supervised (labelled data)
Classification: categorical target
Regression (estimation): numeric target
Algorithms:
Logistic regression
Decision tree classifier
KNN
Performance:
Accuracy, Precision, Recall, Sensitivity, Specificity, F1-score, TPR, FPR, confusion matrix, ROC, AUC
Algorithms:
Linear regression
Decision tree regressor
KNN
Performance:
MAE, MSE, RMSE,
R^2


Myelin sheath
Axon
ANN and deep learning
MLP classifier and regressor
Concept illustration Example

Supervised Machine Learning with KNN

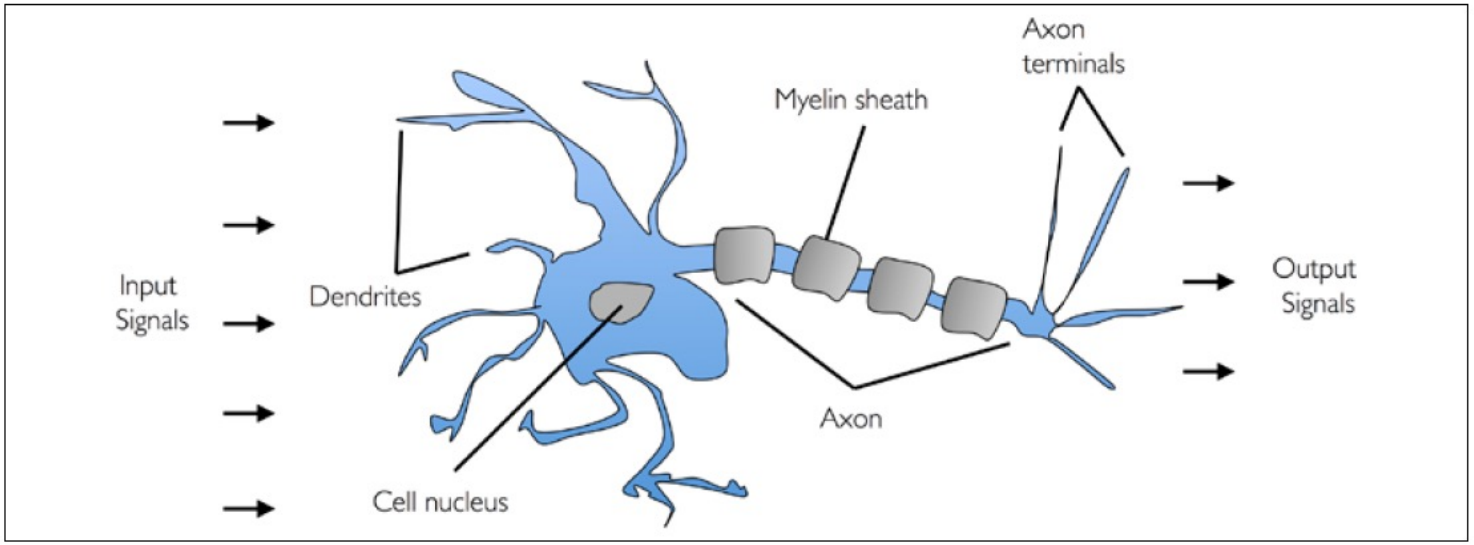




http://ebookcentral.proquest.com/lib/deakin/detail.action?docID=6005547

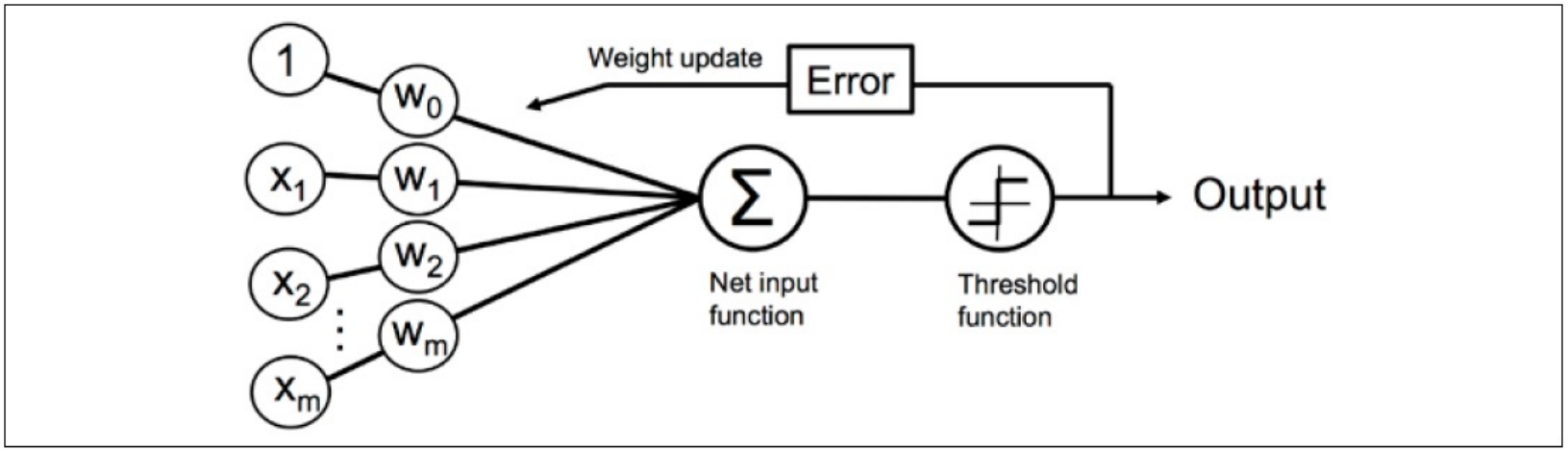
From a description of the simplified brain cell

Raschka and Mirjalili (2019)



- McCulloch and Pitts' description of the simplified brain cell (1943)

To concept of the perceptron learning rule



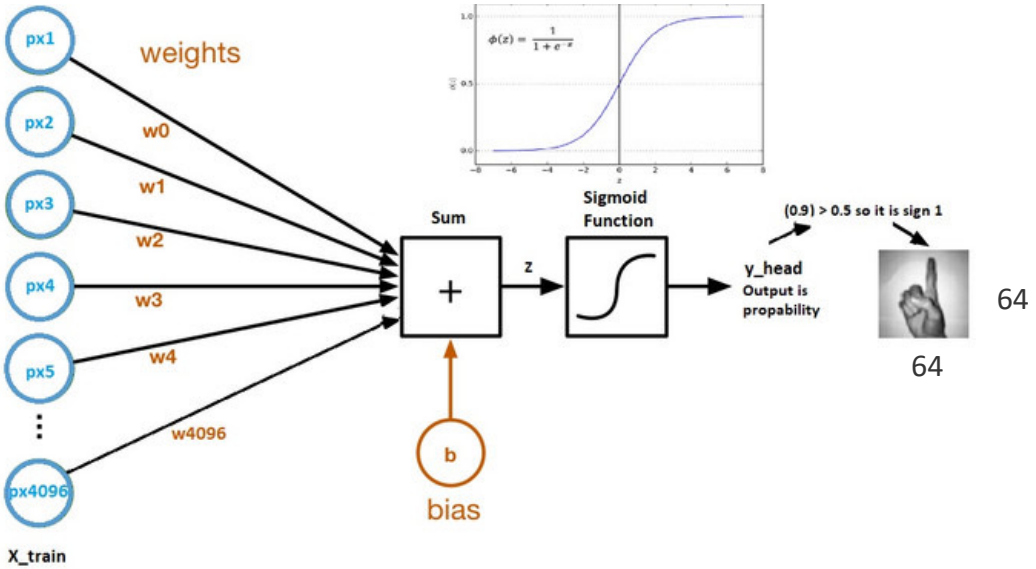
Raschka and Mirjalili (2019)

- Frank Rosenblatt's concept of the perceptron learning rule (1957)

Most basic ANN

- The most basic neural network is a single neuron with a linear activation function, functioning like a linear regression model for supervised regression.

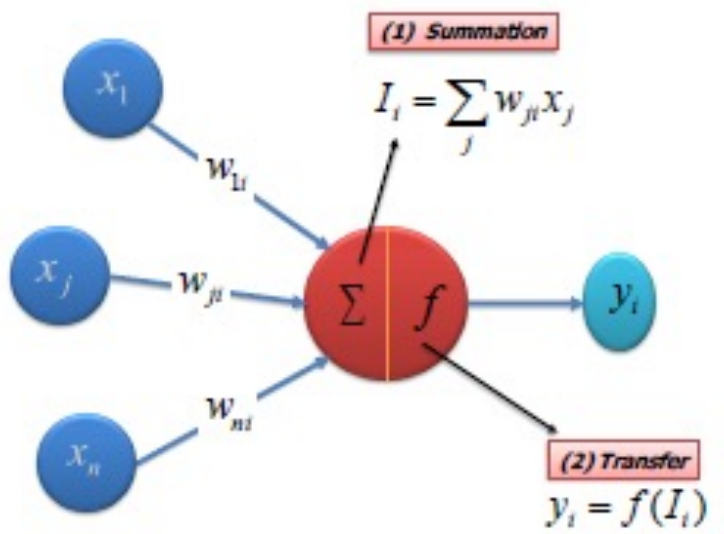
- The sigmoid activation function can be added to convert outputs to probabilities, making it suitable for binary classification



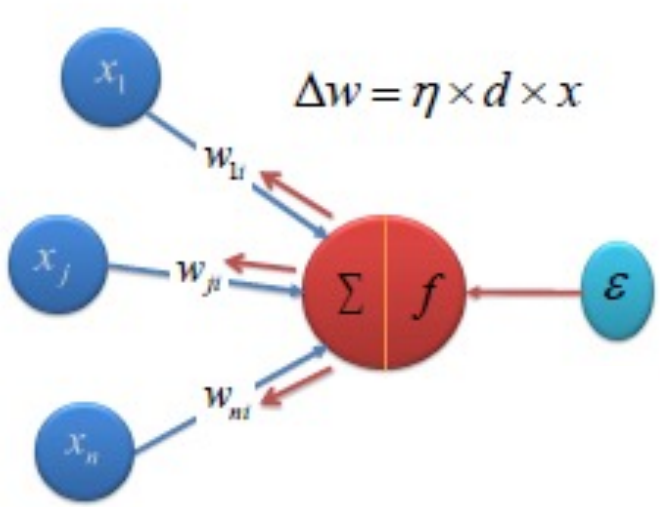
https://www.analyticsvidhya.com/blog/2021/05/a-comprehensive-tutorial-on-deep-learning-part-1/

Forward propagation and backward propagation processes

Feedforward Input Data



Backward Error Propagation



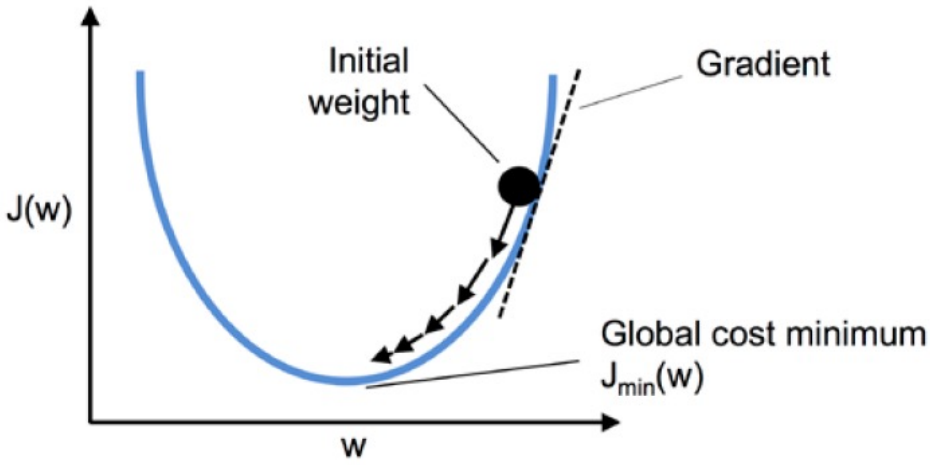
https://www.saedsayad.com/artificial_neural_network.htm

The cost function J(w)

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right)^2$$

sum of squared errors (SSE) between the calculated outcome and the true class label

1/2 added for convenience to make it easier to derive the gradient of the cost or loss function

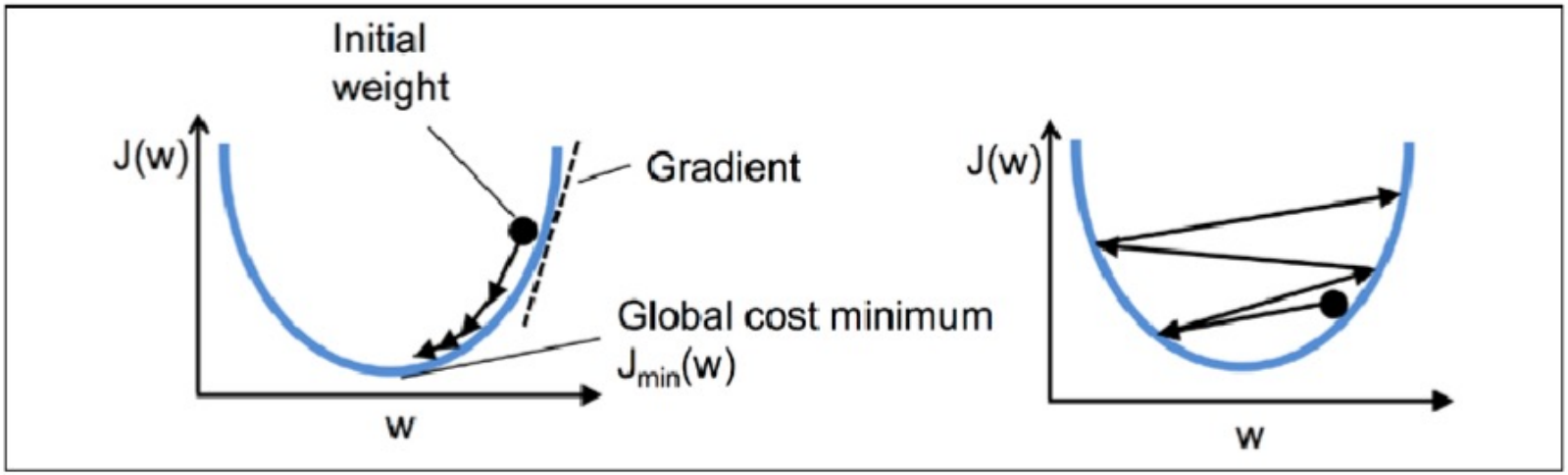


gradient descent is an optimisation algorithm to find the weights that minimise our cost function

$$\Delta w_j = \eta \sum_i \left(y^{(i)} - \phi\left(z^{(i)}\right)\right) x_j^{(i)}$$

Raschka and Mirjalili (2019)

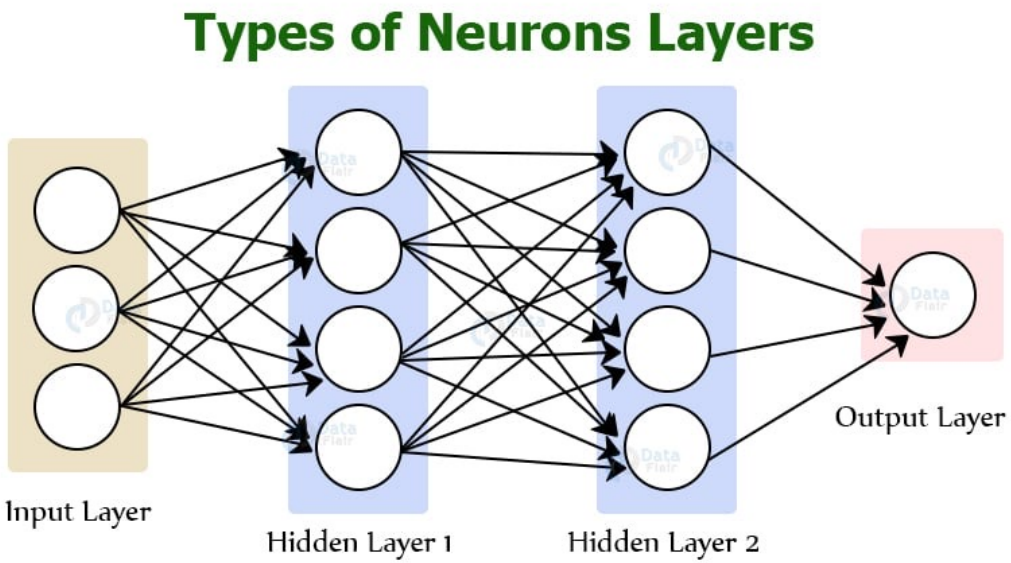
Learning rates and epochs



- epochs #passes through training set
- η learning rate

Raschka and Mirjalili (2019)

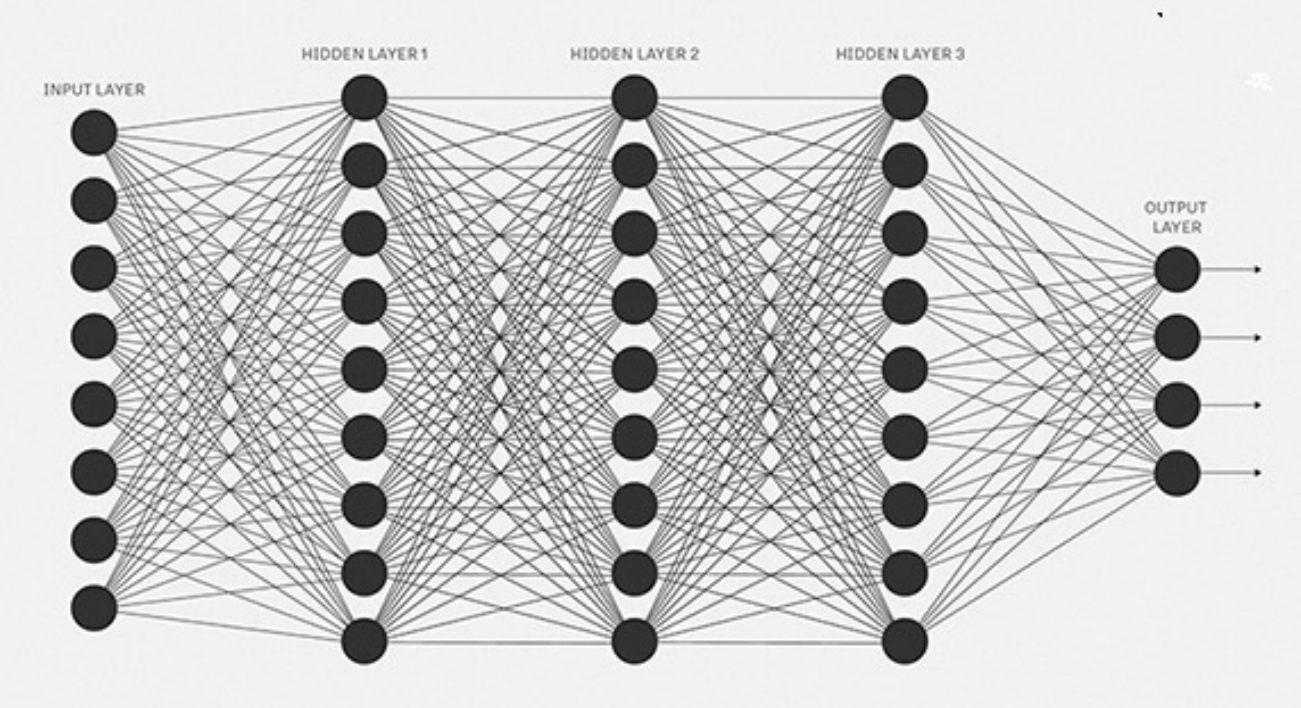
Multilayer perceptron



One output: binary classification

https://data-flair.training/blogs/how-deep-learning-works/

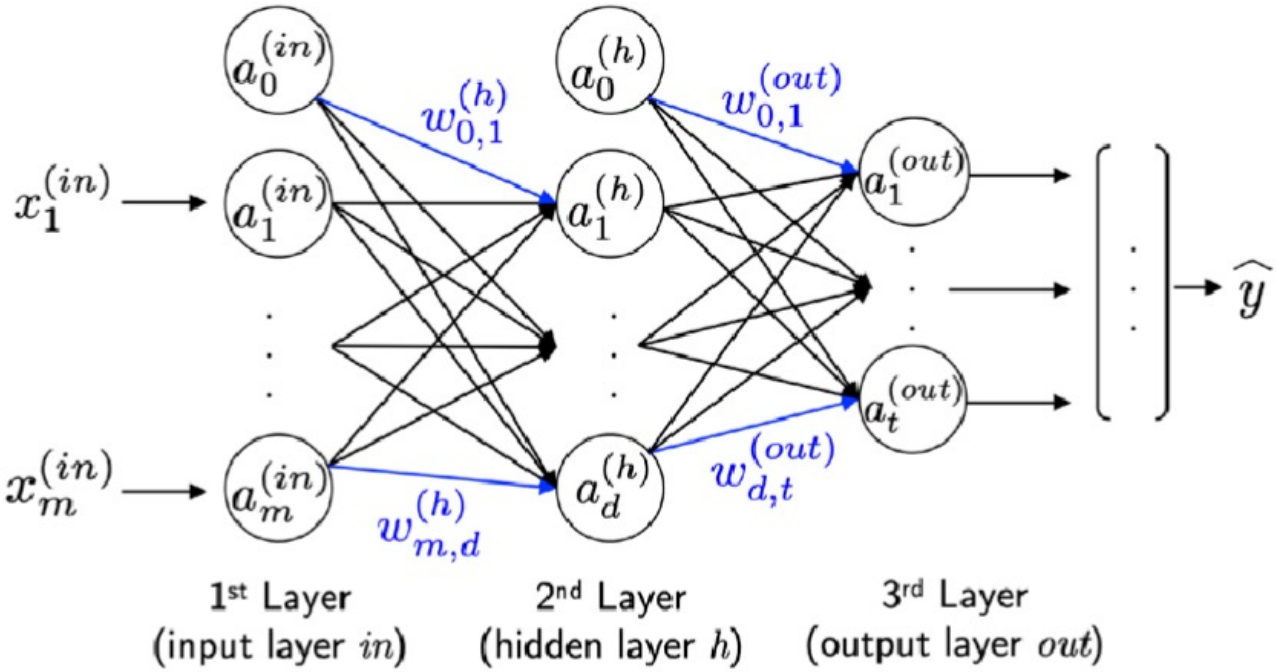
Multilayer perceptron



Multiple outputs: multiclass classification (OvR)

https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

Multilayer perceptron



One output: binary classification

Multiple outputs: multiclass classification (OvR)

Raschka and Mirjalili (2019)

Common Activation Functions

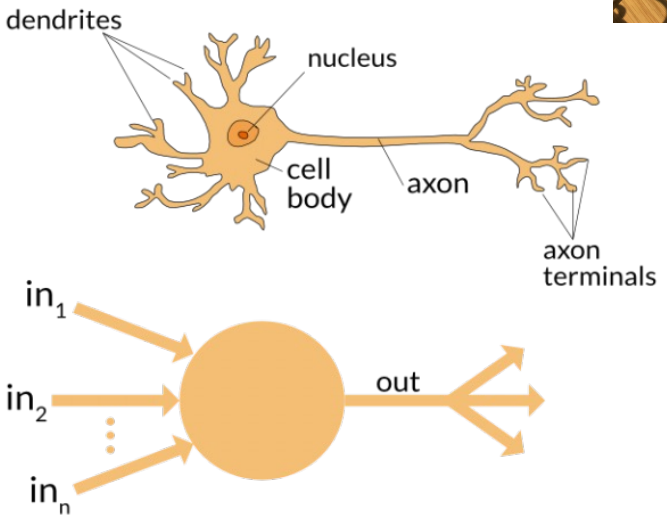
- **Linear Activation Function**: Directly outputs the weighted sum of inputs.
- **ReLU (Rectified Linear Unit) Function**: Outputs 0 for negative inputs and the input itself for positive inputs. ReLU is commonly used in hidden layers of deep networks due to its ability to mitigate the vanishing gradient problem.
- **Softmax function** transforms the raw scores (logits) into probabilities that sum to 1, to output a probability distribution over multiple classes for multiclass Classification.
- **Sigmoid Function**: Outputs a value between 0 and 1. It can be used in the output layer for binary classification to interpret the output as a probability.
- **Hyperbolic Tangent (Tanh) Function**: Outputs a value between -1 and 1. It's often used in hidden layers to normalize the output to a range centred around zero, which can help with training stability.

Activation functions are a hyperparameter in designing ANNs

- **Classifier:**
  - 'logistic' (sigmoid) producing probabilities between 0 and 1
  - softmax for multi-class classification
- **Regressor:**
  - a linear activation function in the output layer by default
- **Hidden layers:**
  - 'relu' avoiding the vanishing gradient problem and enabling faster convergence.

Artificial neural network (ANN)

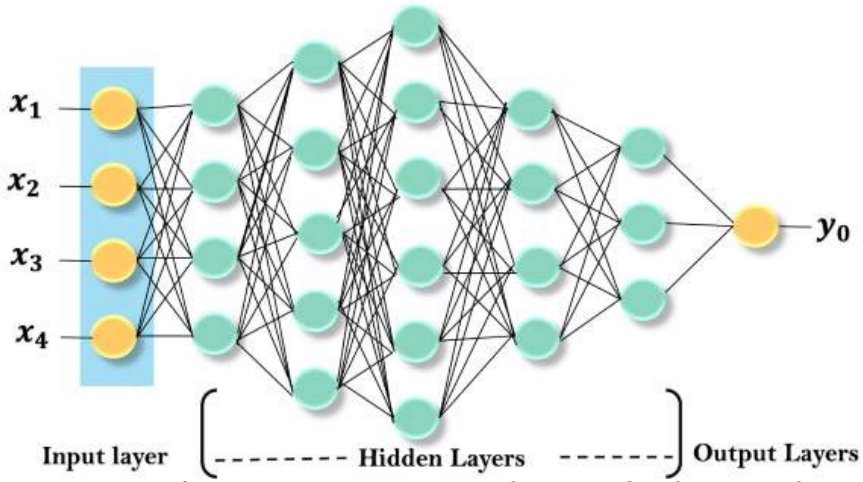
- A type of machine learning model that is inspired by the structure and function of the human brain.
- Composed of interconnected nodes, called neurons.
- Each neuron receives input from other neurons or external sources, applies a mathematical operation to the input, and outputs a signal to other neurons.
  - Summation: a linear transformation using the weights and biases = weighted sum of inputs + bias
  - Transfer: a non-linear transformation of the summation result e.g. Rectified Linear Unit ReLU(x) = max(0,x)
- Deep learning is based on artificial neural networks with multiple layers and a large amount of data.



https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

Deep Learning example

- Patient data and clinical data are specified as inputs x_1, x_2, x_3... (age, gender, med. compliance, #procedures, LoS, type of visit, #emergency_visits, #prior_admissions, etc)
- Unplanned readmission risk score can be specified as the label output (y_0)
- Neural network models can be trained, tested and selected for use in predict readmission for new visits/discharges



https://static.javatpoint.com/tutorial/ai/images/deep-learning-works.png

For example

Journal of Biomedical Informatics 97 (2019) 103256

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

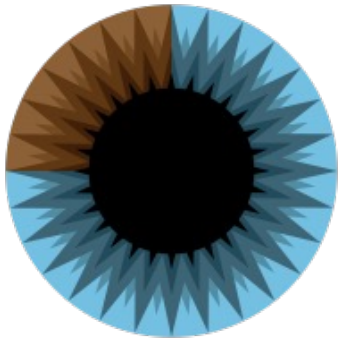
journal homepage: www.elsevier.com/locate/yjbin

Readmission prediction using deep learning on electronic health records

Awais Ashfaq[a,b,*], Anita Sant'Anna[a], Markus Lingman[b,c], Sławomir Nowaczyk[a]

[a] Center for Applied Intelligent Systems Research, Halmstad University, Sweden
[b] Halland Hospital, Region Halland, Sweden
[c] Institute of Medicine, Dept. of Molecular and Clinical Medicine/Cardiology, Sahlgrenska Academy, University of Gothenburg, Sweden

Concept illustration Example

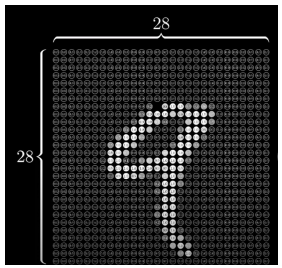


Grant Sanderson
3Blue1Brown
https://youtu.be/aircAruvnKk

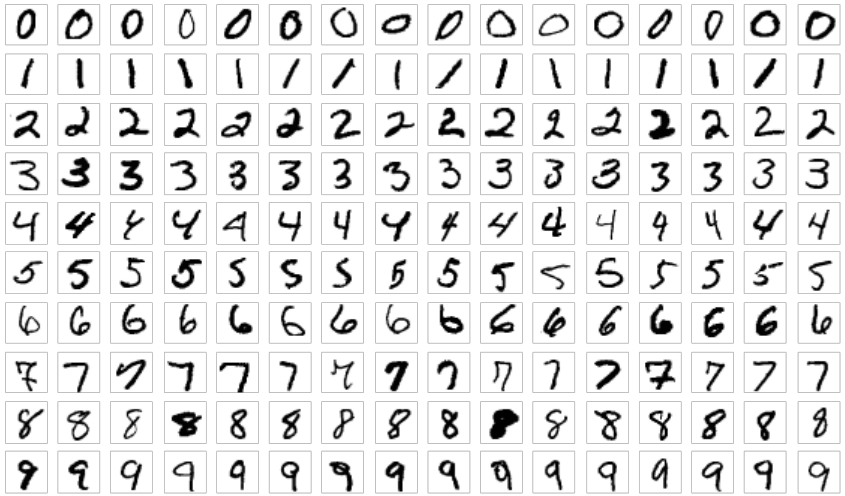
MNIST Handwritten Digit Classification Dataset

Modified National Institute of Standards and Technology.

60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9.



Goal: Classify a given image of a handwritten digit into one of 10 classes representing a digit from 0 to 9.

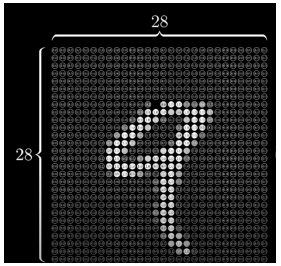


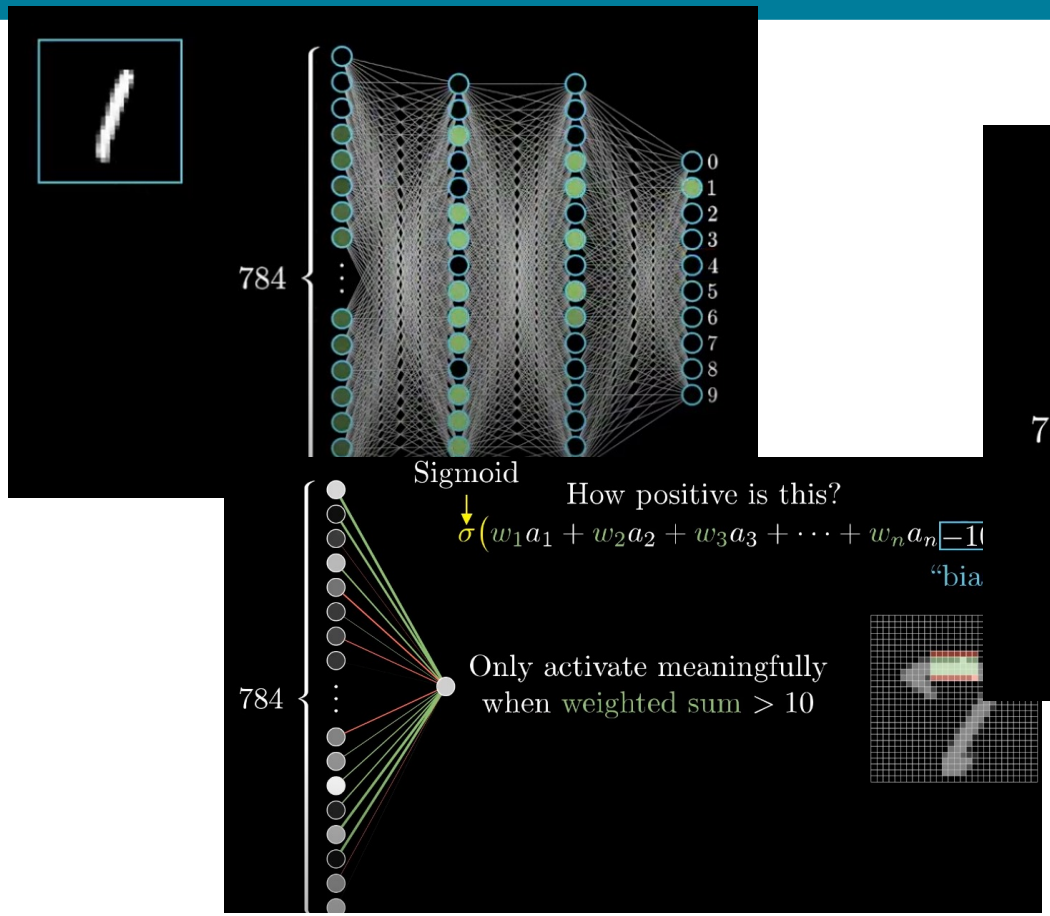
https://upload.wikimedia.org/wikipedia/commons/2/27/MnistExamples.png

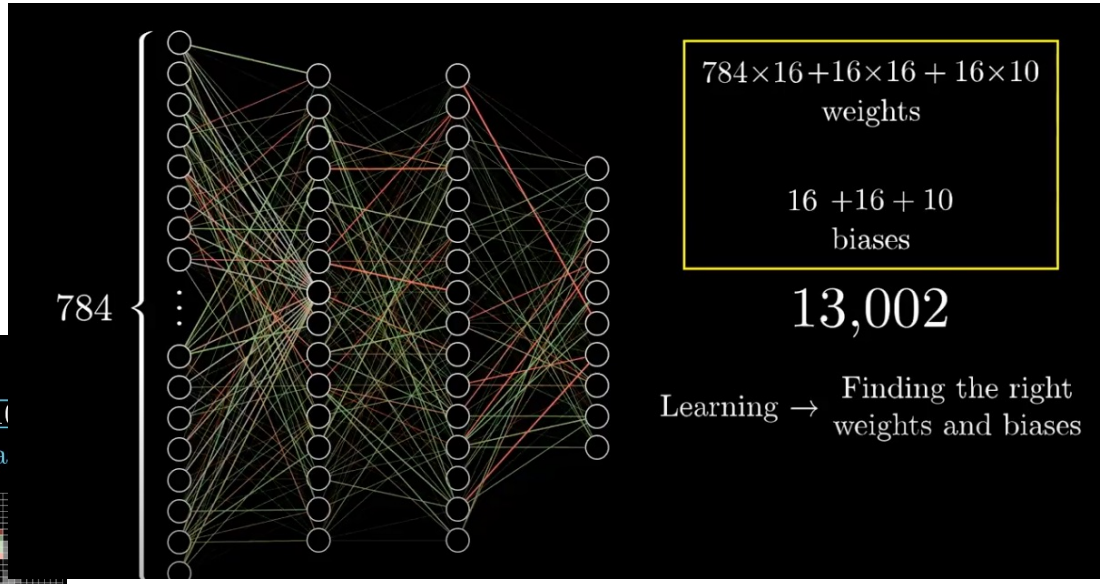
Deep Learning - example

https://youtu.be/aircAruvnKk

3Blue1Brown

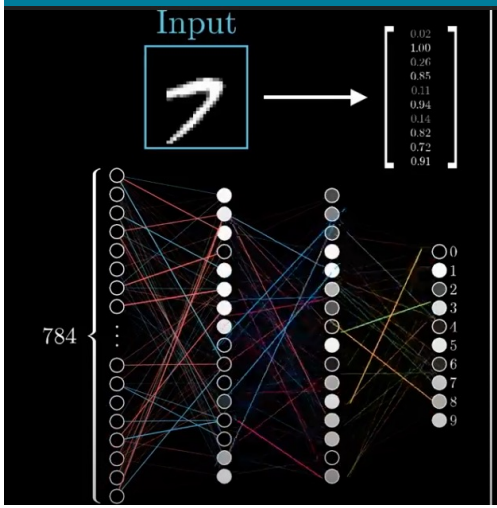






https://www.youtube.com/watch?v=IHZwWFHWa-w&t=329s

Deep Learning – How does it learn?

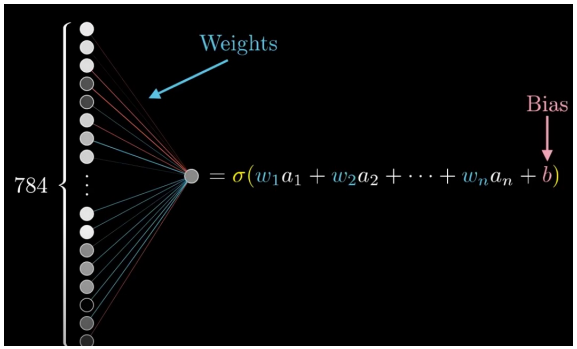


Neural network function

Input: 784 numbers (pixels)

Output: 10 numbers

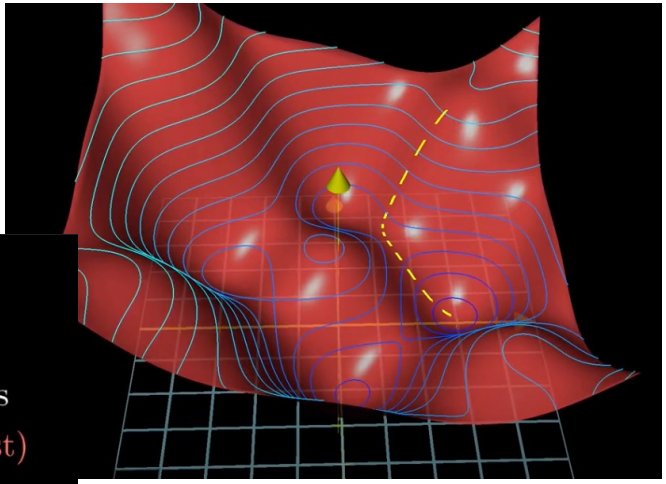
Parameters: 13,002 weights/biases

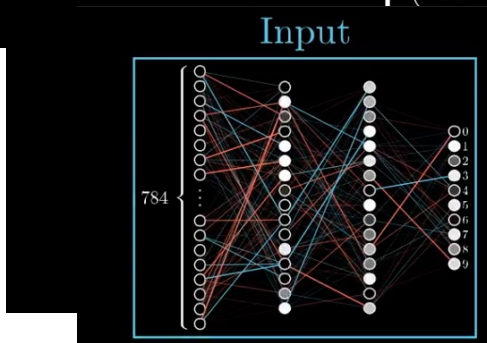


Average cost of all training data...

$$\begin{bmatrix} (0.03 - 0.00)^2 + \\ (0.92 - 0.00)^2 + \\ (0.97 - 0.00)^2 + \end{bmatrix}$$

What's the "cost" of this difference?





Cost: 5.4

Cost function

Input: 13,002 weights/biases

Output: 1 number (the cost)

Parameters: Many, many, many training examples

https://youtu.be/aircAruvnKk

Pros and cons and beyond

Pros

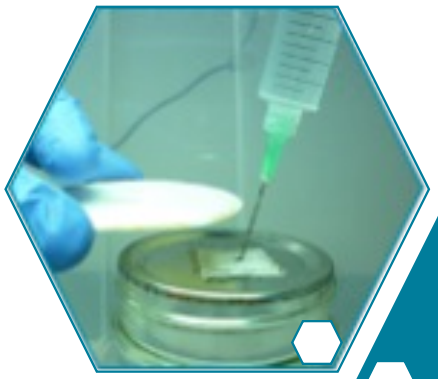
- High accuracy
- Can automatically learn relevant features from raw data (instead of manual feature engineering)
- Scale to large datasets and high-dimensional data
- A wide range of applications: text, image, audio

Cons

- Require a lot of computing power and time-consuming
- Require large amounts of high-quality data to learn effectively
- May overfit to the training data
- **Lack of Interpretability: black boxes**
- Can be difficult to implement as expertise is required to optimise the model

Scale data

LeCun et al. (2015, 1998); Goodfellow, et al., 2016; Schmidhuber 2015



MLP classifier

ML in Business Framing

Business needs

- Long delay in returning pathology results
- Novice pathologists need training

Data

- Sources: Past biopsy data and results
  - Internal and external
- **Data content: Biomarkers and diagnoses**

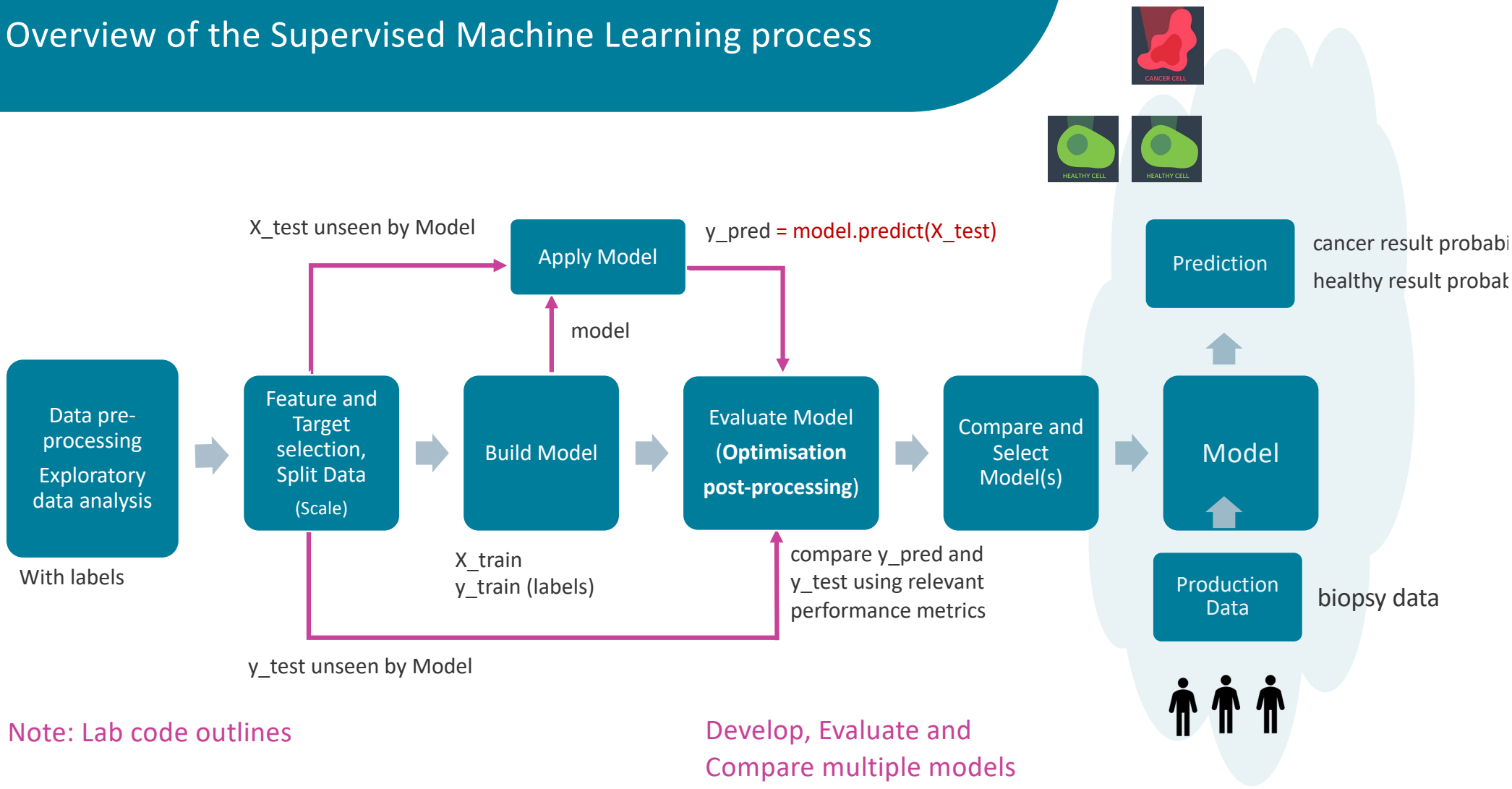
Analytics (ML)

- **Supervised ML, classification**
- Build and test a logistic regression model that best fit the historical biomarkers and diagnoses

Deployment

- Decision augmentation: in interpreting biopsy data -> reduced time and improved accuracy
- Training novice pathologists

Overview of the Supervised Machine Learning process



Note: Lab code outlines

Develop, Evaluate and Compare multiple models

Dataset

Data:
V1, V2, V7, V8, V9: biological variables
Diagnosis: healthy or cancerous

Sample size: 699
Number of columns: 7
V1, V2, V7, V8, V9: integer
Label: diagnosis: string

Source: **adapted** from a dataset provided by Dr Mark Griffin, Industry Fellow, University of Queensland

dependent varial
label (y)

ID	V1	V2	V7	V8	V9	diagnosis
1177399	8	3	1	6	2	healthy
1246562	10	2	1	1	2	healthy
1108370	9	5	2	1	5	healthy
1165926	9	6	2	9	10	healthy
1167439	2	3	2	5	1	healthy
1226012	4	1	2	1	1	healthy
601265	10	4	2	3	1	healthy
1295186	10	10	2	8	1	healthy
1343068	8	4	2	5	2	healthy
1065726	5	2	3	6	1	healthy
1102573	5	6	3	1	1	healthy
1106829	7	8	3	8	2	healthy
1108449	5	3	3	4	1	healthy
1111249	10	6	3	6	1	healthy
1112209	8	10	3	9	1	healthy
1116116	9	10	3	3	1	healthy
1116132	6	3	3	9	1	healthy
1126417	10	6	3	2	3	healthy
1166654	10	3	3	10	2	healthy
1169049	7	3	3	2	7	healthy
1171845	8	6	3	1	1	healthy

[skip data loading, preprocessing, EDA.]

```
#import train_test_split, StandardScaler and MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.neural_network import MLPClassifier

#import classes to display RocCurve and Confusion Matrix
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import ConfusionMatrixDisplay
```

Data Splitting and Normalisation

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y,
random_state=2023) # 70% training and 30% testing
```

All predictors are of the same range

scikit-learn's MLP and Keras

- **popular libraries for building and training ANNs in Python.**
- **API:**
  - Scikit-learn's MLP has a simpler API, thus easier to use for beginners
  - Keras has a more complex and flexible API, requires more expertise and knowledge of deep learning concepts
- **Architecture:**
  - Scikit-learn's MLP supports only fully connected feedforward neural networks.
  - Keras supports a wider range of architectures including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more.
- **Performance:**
  - Scikit-learn's MLP may not run fast on large datasets
  - Keras is built on top of TensorFlow, which allows you to use GPUs and TPUs for faster computation when working with large datasets or complex models.

Sci-learn MLP classifier

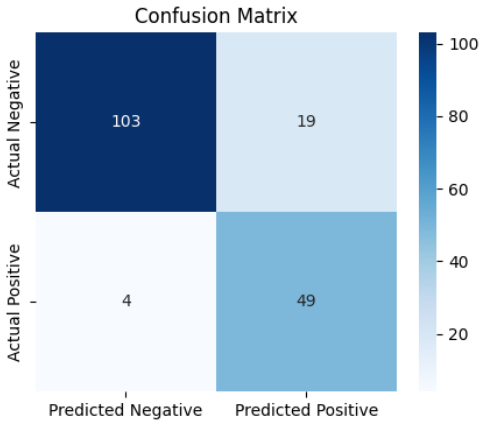
```
# create an MLP classifier with 2 hidden layers
ann_clf = MLPClassifier(hidden_layer_sizes=(32, 16), max_iter=1000,
random_state=2023, early_stopping=True)

# train the classifier on the training data
ann_clf.fit(X_train, y_train)



# evaluate the classifier on the testing data
y_pred = ann_clf.predict(X_test)

#get predicted probabilities for the main class
y_pred_probs = ann_clf.predict_proba(X_test)
y_pred_probs = y_pred_probs[:, 1]
```

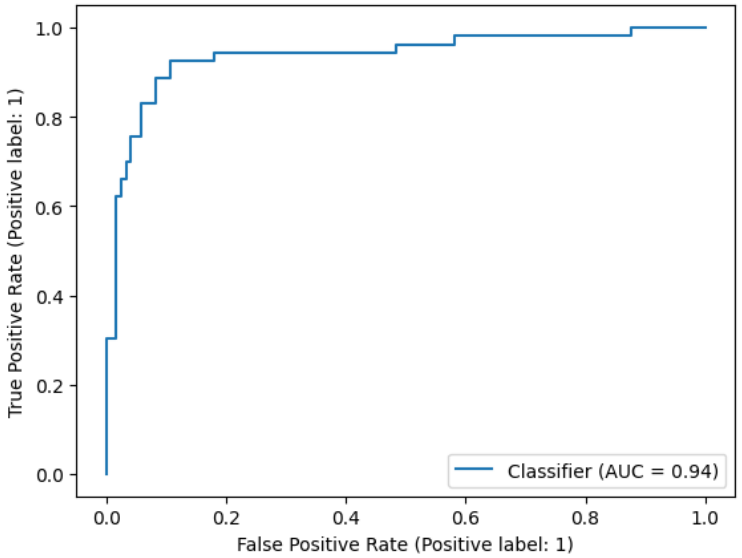
Test the ANN model

```
[[103  19]
 [  4  49]]
              precision    recall  f1-score   support

           0       0.96      0.84      0.90       122
           1       0.72      0.92      0.81        53

    accuracy                           0.87       175
   macro avg       0.84      0.88      0.85       175
weighted avg       0.89      0.87      0.87       175
```

```
#print confusion matrix and evaluation report
cm=confusion_matrix(y_test, y_pred)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
#RocCurveDisplay.from_estimator(logreg,X_test, y_test)
RocCurveDisplay.from_predictions(y_test, y_pred_probs)
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.show()
```





Keras MLP

- Sequential allows users to create models layer-by-layer; including dense layers.
- Dense is a type of layer in Keras to create fully connected neural networks. The output of a dense layer is obtained by applying an activation function to a weighted sum of the inputs.
- EarlyStopping is a callback function in Keras to stop the training of a neural network if the validation loss stops improving for a certain number of epochs. It helps to prevent overfitting and improve the generalization of the model.

Keras MLP: Example

```
# Construct the ANN model structure
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(X_train_scaled.shape[1],)))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])



# Train model
early_stopping = EarlyStopping(patience=5, monitor='val_loss')
model.fit(X_train, y_train, validation_split=0.2, epochs=10, batch_size=32,
        callbacks=[early_stopping])
```

Keras MLP: Example

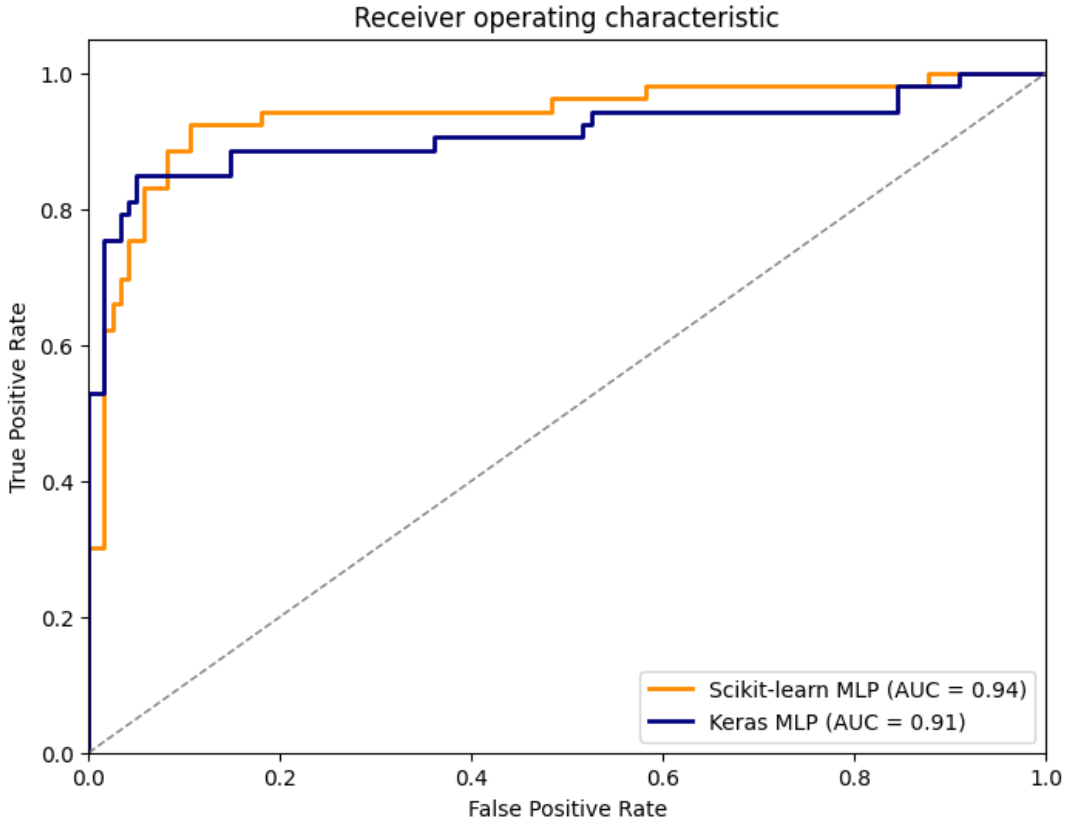
```
[[116   6]
 [ 10  43]]
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       122
           1       0.88      0.81      0.84        53

    accuracy                           0.91       175
   macro avg       0.90      0.88      0.89       175
weighted avg       0.91      0.91      0.91       175
```

```
# Predict using the Keras model and calculate performance metrics
# Predict test data labels
y_pred_probs_keras = model.predict(X_test)
y_pred_keras = (y_pred_probs_keras > 0.5)


#print confusion matrix and evaluation report
from sklearn.metrics import classification_report, confusion_matrix
cm_keras=confusion_matrix(y_test, y_pred_keras)
print(cm_keras)
print(classification_report(y_test, y_pred_keras))
```

Model comparison

Performance Metrics	Scikit-learn MLP Classifier	Keras MLP
Accuracy	0.87	0.91
Precision	0.72	0.88
Recall	**0.92**	0.81
F1 score	0.81	0.84

```
Scikit-learn MLP AUC: 0.94
Keras MLP AUC: 0.91
```



MLP regression

ML in Business Framing: Estimating Health Insurance Premium

- Health insurance company introducing a self-service to support prospective policy holders' decision making and improve their experience; reduce #staff hours to answer quote enquiries
- Need to estimate insurance premium

Business needs (BA)

- Sources:
  - Internal past data: policy dataset
- **Data content**:
  - Policy holder's demographics: age, sex, dependants, region
  - Policy holder's health behaviours: BMI, smoker
  - Label: charger label

Data

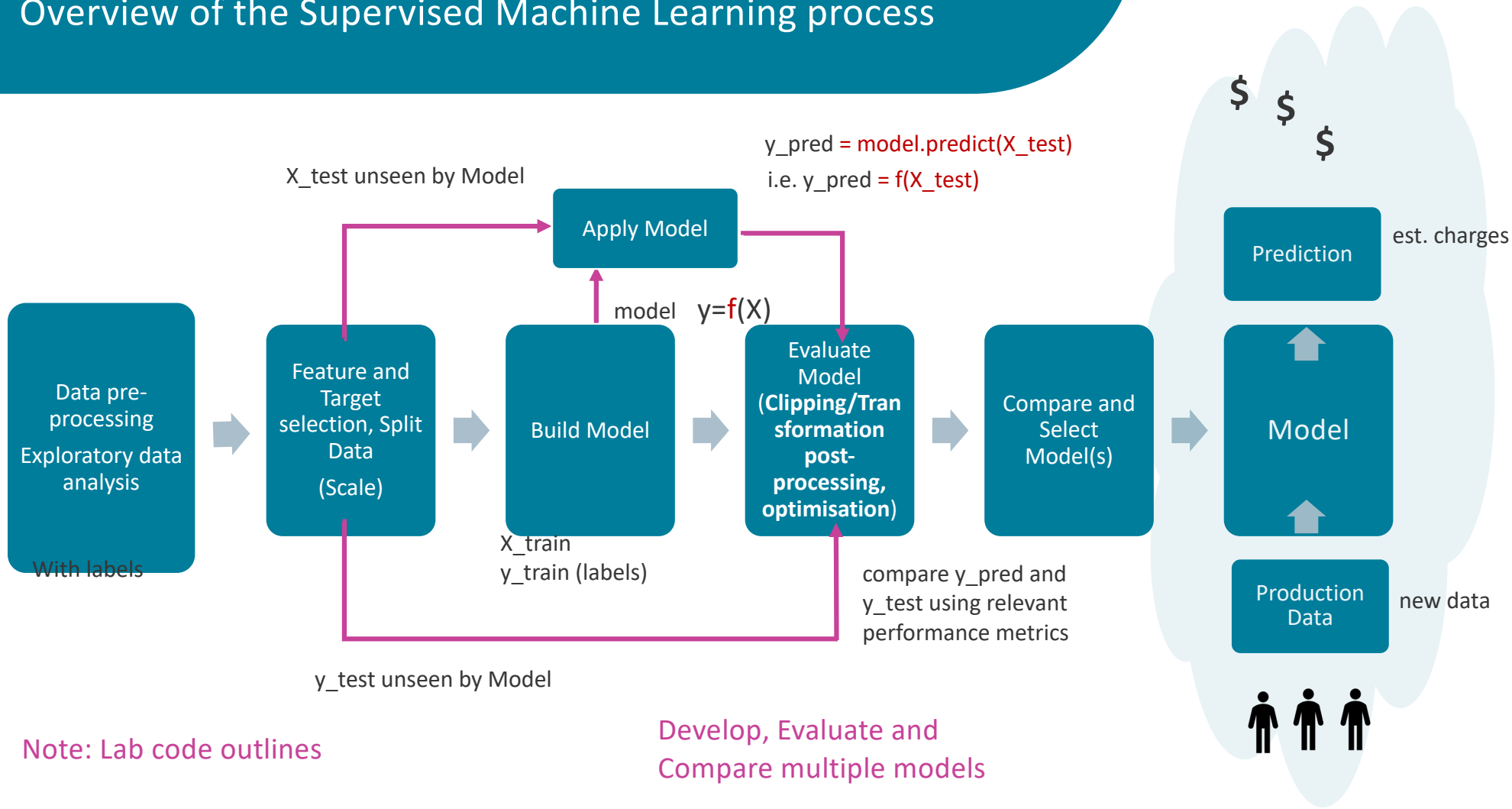
- **Supervised ML, estimation**
- Build and test a **KNN** model that best fit the existing policy holders demographics, health behaviours, and premium values

Analytics (ML)

- Get a Quote self-service tool on website

Deployment (Serving)

Overview of the Supervised Machine Learning process



Note: Lab code outlines

Develop, Evaluate and Compare multiple models

Feature selection and data normalisation

```
X=records.drop('charges', axis=1)
y=records['charges']

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2023)


# Scale the features using Min-Max Scaling
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model building

```
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error


# Initialise an ANN model
ann_reg = MLPRegressor(hidden_layer_sizes=(32,16), activation='relu',
solver='adam', max_iter=1000, random_state=2023)

# Fit it to the scaled training data
ann_reg.fit(X_train_scaled, y_train)
```

Model testing and **post-processing**

```
# Use the trained model to predict on the scaled test data
y_pred_mlp= ann_reg.predict(X_test_scaled)
```

Clipping a numeric outcome within a range

```
# For example, we can set negative or small predictions to a base insurance premium,
# e.g. 1131.5 (min) from the y_test set
y_pred_mlp = np.maximum(y_pred_mlp, 1131.5)
```

To ensure that all outputs are non-negative

```
# e.g. 1131.5 (min) from the y_test set
y_pred_mlp = np.maximum(y_pred_mlp, 0)
```

Check: business rules, domain knowledge; and existing datasets

Activation function **ReLU (Rectified Linear Unit)** f(x) = max(0, x)

Clipping a numeric outcome within a range

```
# Clip the 'Score' column to be between 0 and 100
records['Score'] = records['Score'].clip(lower=0, upper=100)
```

```
# Clip the 'Score' column using NumPy
records['Score'] = np.clip(records['Score'], a_min=0, a_max=100)
```

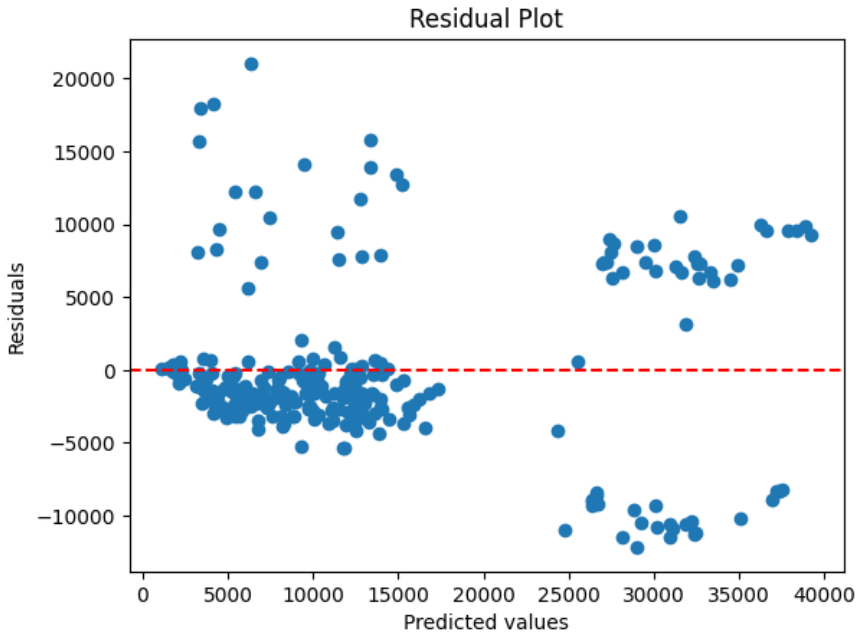
Check: business rules, domain knowledge; and existing datasets

Calculate Outliers and Clip the Values

```
# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = records['Score'].quantile(0.25)
Q3 = records['Score'].quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

# Define lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Clip the 'Score' column using the calculated bounds
records['Score'] = records['Score'].clip(lower_bound, upper_bound)
```

Model evaluation

```
Scikit learn MLP RMSE: 5699.234
Scikit learn MLP MAE: 4029.807
Scikit learn MLP R-squared: 0.753
```




```
# Calculate RMSE, R-squared, and MAE
rmse_mlp = np.sqrt(mean_squared_error(y_test, y_pred_mlp))
r2_mlp = r2_score(y_test, y_pred_mlp)
mae_mlp = mean_absolute_error(y_test, y_pred_mlp)

# Print the evaluation metrics
print("Scikit learn MLP RMSE: {:.3f}" .format(rmse_mlp))
print("Scikit learn MLP MAE: {:.3f}" .format(mae_mlp))
print("Scikit learn MLP R-squared: {:.3f}" .format(r2_mlp))
```

Keras MLP

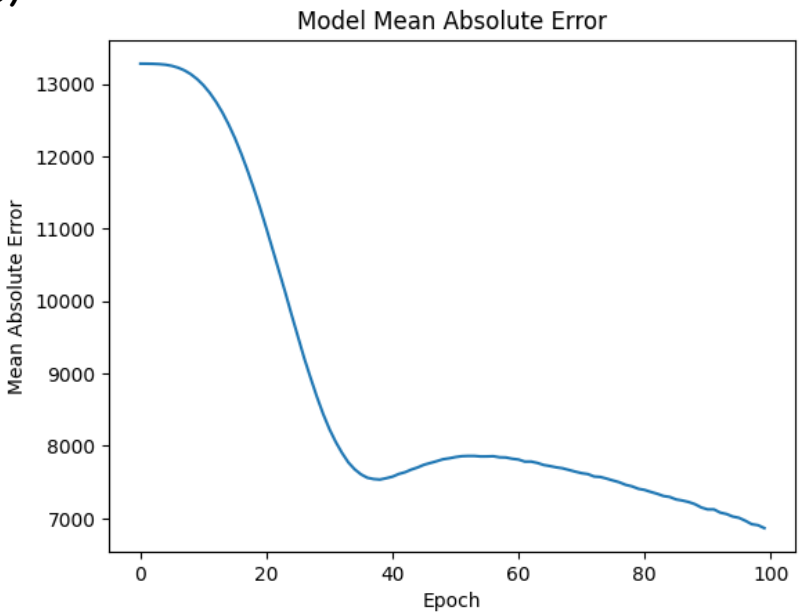
```
# construct the keras model
mlp_keras = Sequential()
mlp_keras.add(Dense(32, input_dim=6, activation='relu', input_shape=(X_train_scaled.shape[1],)))
mlp_keras.add(Dense(16, activation='relu'))
mlp_keras.add(Dense(1, activation='linear'))

# compile the keras model
mlp_keras.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squared_error'])

# fit the keras model on the dataset
history = History()
history=mlp_keras.fit(X_train_scaled, y_train, epochs=100, batch_size=32, verbose=0,
callbacks=[history])
```

Keras MLP

```
# make predictions using the model
y_pred_keras = mlp_keras.predict(X_test_scaled)

# calculate evaluation metrics
rmse_keras = np.sqrt(mean_squared_error(y_test, y_pred_keras))
mae_keras = mean_absolute_error(y_test, y_pred_keras)
r2_keras = r2_score(y_test, y_pred_keras)
```

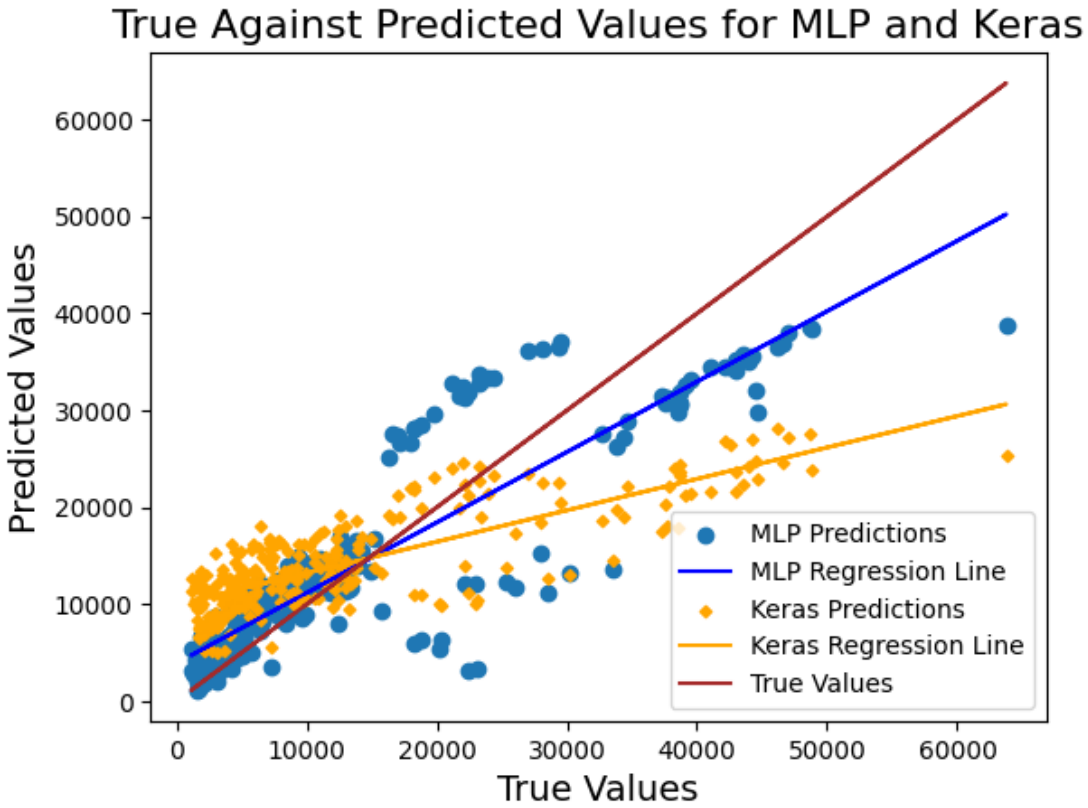
```
9/9 [==============================] - 0s
3ms/step
RMSE: 5678.51
MAE: 4161.28
R-squared: 0.79
```

```
plt.plot(history.history['mean_squared_error'])
plt.title('Model Mean Squared Error')
plt.ylabel('Mean Squared Error')
plt.xlabel('Epoch')
plt.show()
```



Model comparison

Performance Metrics	Scikit-learn MLP Regressor (32,16)	Keras MLP (32, 16, 1)
RMSE	5699.164	8363.66
MAE	4028.447	6766.44
R^2	0.753	0.47

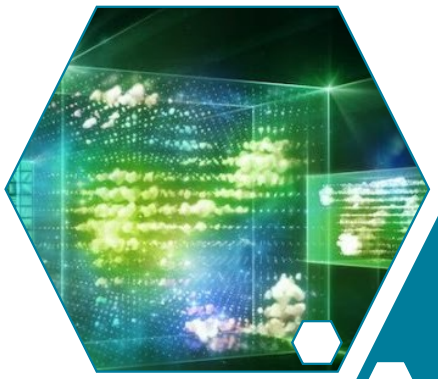


Model comparison

Performance Metrics	Scikit-learn MLP Regressor (32,16)	Keras MLP (32, 16, 1)	KNN (k=31)
RMSE	5699.164	8363.66	6149.137
MAE	4028.447	6766.44	3960.927
R^2	0.753	0.47	0.713

ANN - Recap

- A type of machine learning model that is inspired by the structure and function of the human brain.
- Composed of interconnected nodes, called neurons.
- Neurons perform a linear transformation using the weights and biases = weighted sum of inputs + bias; and a non-linear transformation of the weighted sums
- Each neuron receives input from other neurons or external sources, applies the linear transformation to the input, and outputs a signal to other neurons.
- Deep learning is based on artificial neural networks with multiple layers and a large amount of data.

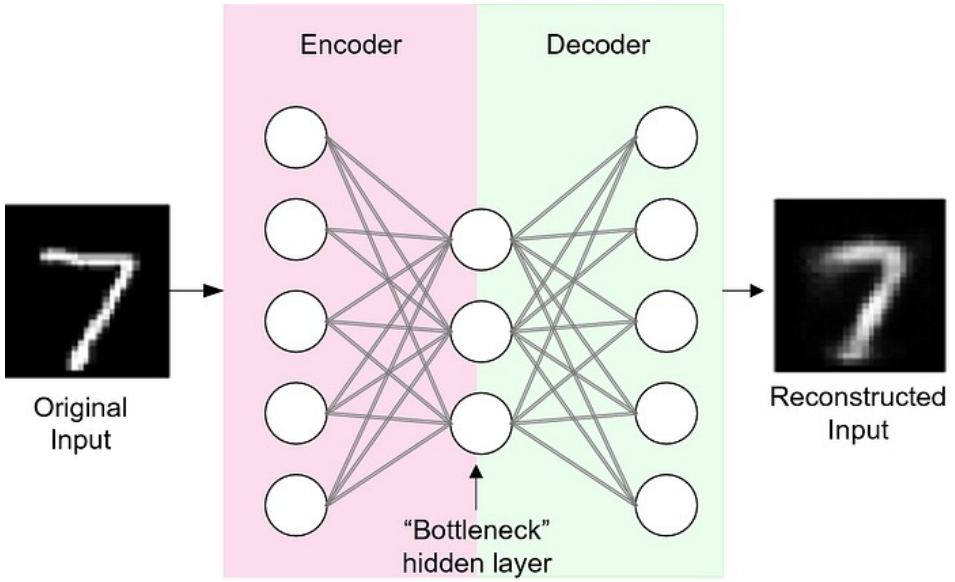


More on ANN

Self-study

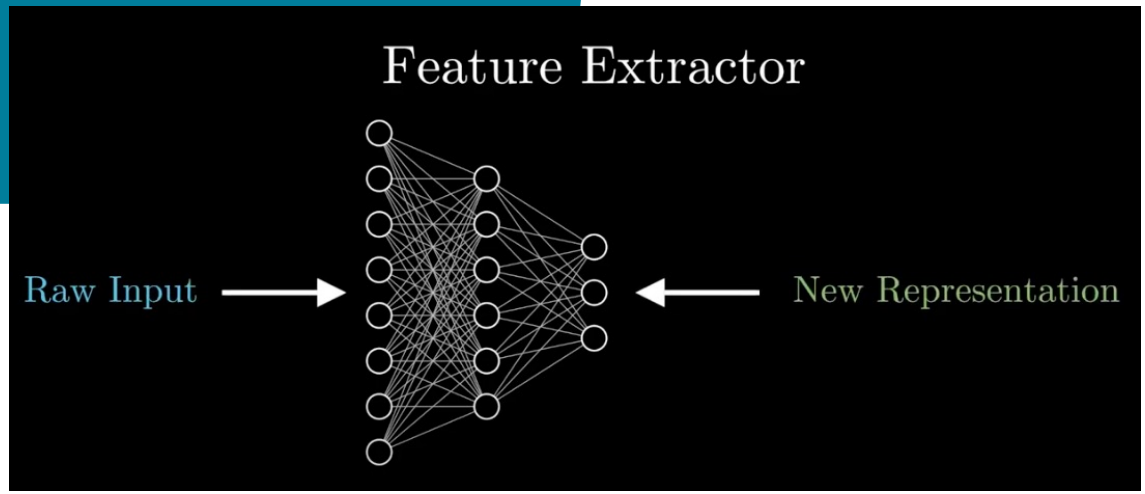
Unsupervised Learning
Autoencoders

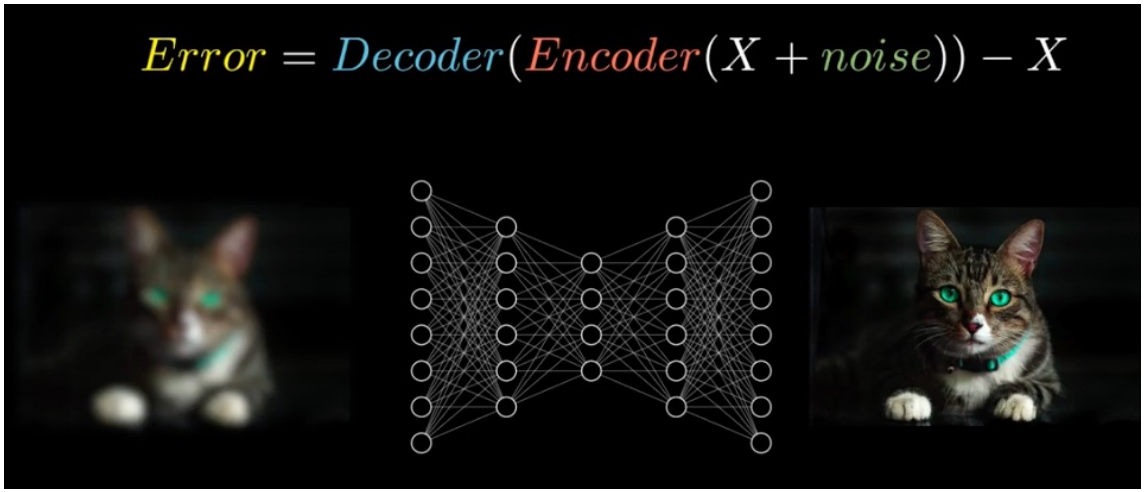
- The encoder maps the input data to a low-dimensional latent space
- The decoder maps the latent space back to the input space.
- Autoencoders aim to reduce the reconstruction errors



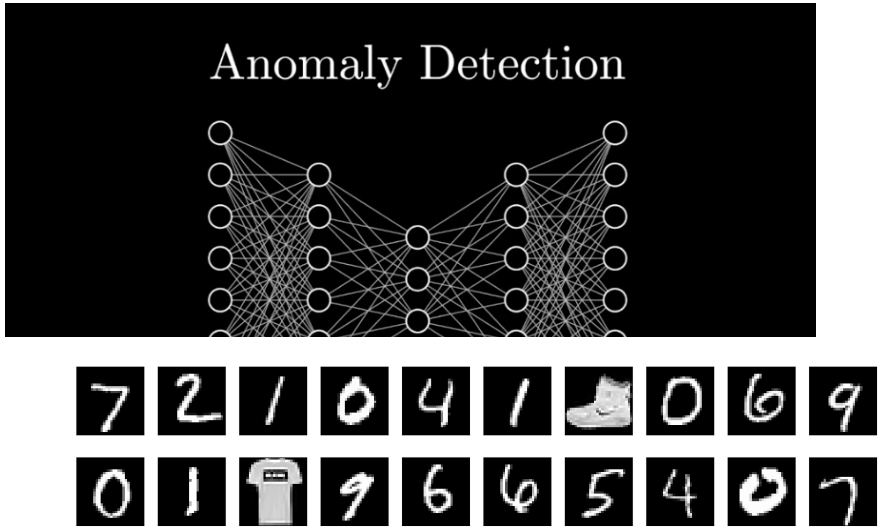
https://medium.com/analytics-vidhya/what-is-auto-encoder-in-deep-learning-5d668f94651b

Applications of Autoencoders



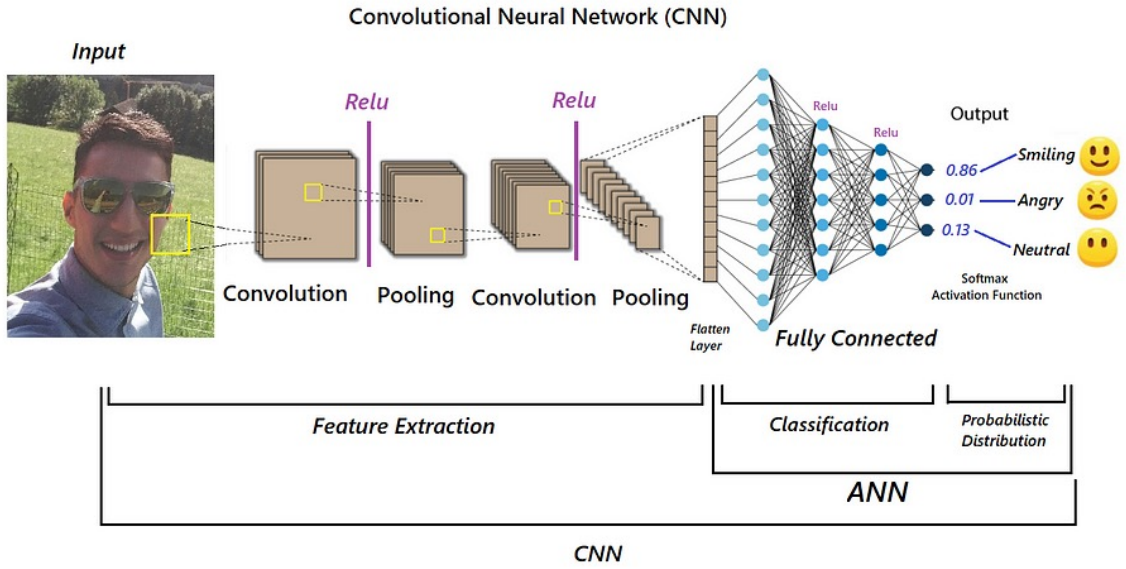


https://www.youtube.com/watch?v=3jmcHZq3A5s



CNN Layers

- Feature extraction:
  - Convolutional layers scan the image using filters to detect features such as edges, corners, and textures.
  - Pooling layers down sample the feature maps to reduce the spatial dimensions of the image,
- Classification:
  - Fully connected (FC) layers use features extracted from the previous layers to classify the image. .
  - The FC layers are trained to recognize patterns in the feature maps and assign probabilities to each classification.



https://medium.com/@El_Fares_Anass/a-better-understanding-how-a-cnn-works-code-available-e880f9a338dc