Rtificial intelligence

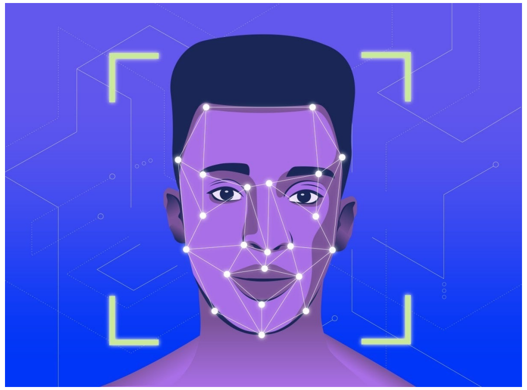
A guide to using R for ML

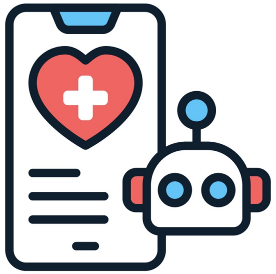
Nikhita Damaraju

06-22-2024

Cascadia R conference 2024

Examples of ML



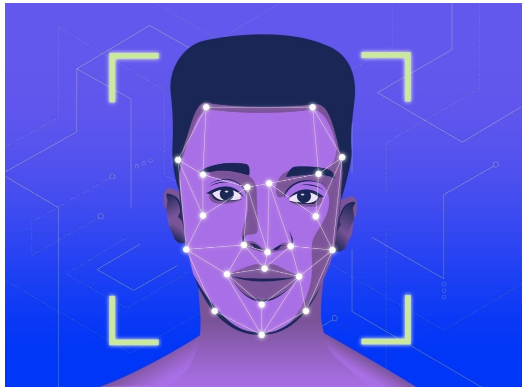


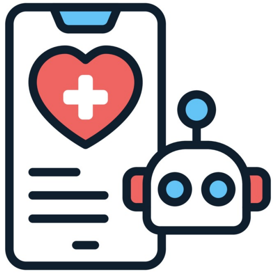
Stock images

When is ML used?

- High-dimensional data

  More features/variables than rows increases complexity of data.





Stock images

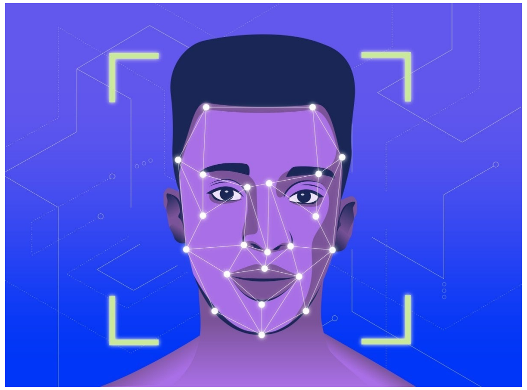
When is ML used?

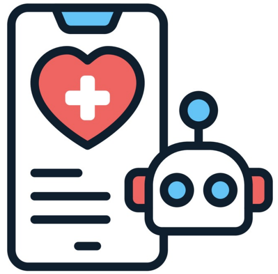
- High-dimensional data

  More features/variables than rows increases complexity of data.

- Non-linear relationships

  Indirect relationship between predictors and outcome.





Stock images

When is ML used?

- High-dimensional data

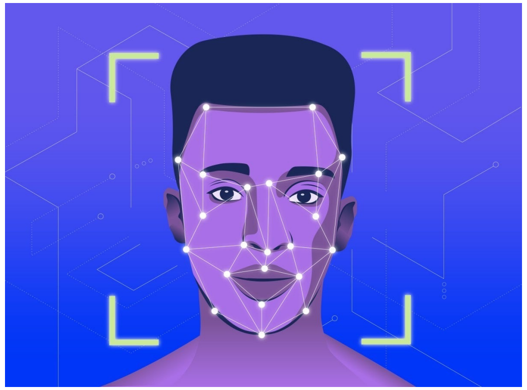
  More features/variables than rows increases complexity of data.

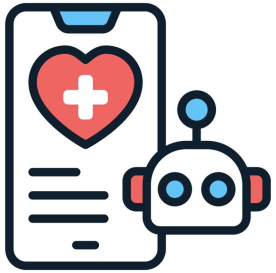
- Non-linear relationships

  Indirect relationship between predictors and outcome.

- Unstructured data

  Non-tabular data like text, images, audio.





Stock images

When is ML used?

- High-dimensional data

  More features/variables than rows increases complexity of data.

- Non-linear relationships

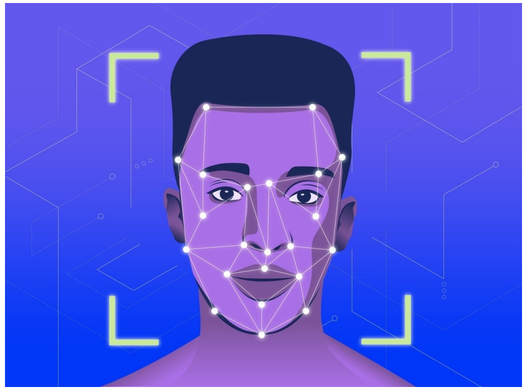
  Indirect relationship between predictors and outcome.

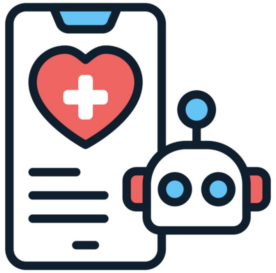
- Unstructured data

  Non-tabular data like text, images, audio.

- Scalability

  Ability to handle millions or billions of data points.





Stock images

Disclaimer!

This talk will not cover:

- R vs python for ML
- How to choose ML methods
- Pros and cons of using R for ML

Building an ML project

Processing data

Picking and training your ML model

Testing the performance of ML models

Redbubble, Adobe stock

Building an ML project

Processing data

Picking and training your ML model

Testing the performance of ML models

Redbubble, Adobe stock

Data processing

- Exploratory data analysis
- Handle missing data
- Outliers
- Feature engineering
- Feature scaling
- Splitting data into training and validation sets

```
# Example
library(dplyr)
library(ggplot2)

# Summary statistics
summary(mtcars)

# Visualization
ggplot(mtcars, aes(x=mpg, y=hp)) + geom_point()
```

Data processing

- Exploratory data analysis
- Handle missing data
- Outliers
- Feature engineering
- Feature scaling
- Splitting data into training and validation sets

mice

stekhoven/
missForest

missForest is a nonparametric, mixed-type imputation method for basically any type of data for the statistical software R.

3 Contributors | 11 Issues | 87 Stars | 23 Forks

```r
# Example
library(mice)
library(missForest)

# Remove rows with missing values
clean_data <- na.omit(mtcars)

# Multiple imputation
imputed_data <- mice(mtcars, m=5, maxit=50, method='pmm', seed=500)

# Imputation using Random Forests
imputed_data_rf <- missForest(mtcars)
```

Packages for imputation in R: https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/

Data processing

- Exploratory data analysis
- Handle missing data
- Outliers
- Feature engineering
- Feature scaling
- Splitting data into training and validation sets

```r
# Example
library(dplyr)
library(ggplot2)

# Identify outliers using boxplot
boxplot(mtcars$mpg)

# Remove outliers
clean_data <- mtcars %>% filter(mpg < quantile(mpg, 0.95))
```

Data processing

- Exploratory data analysis
- Handle missing data
- Outliers
- Feature engineering
- Feature scaling
- Splitting data into training and validation sets

caret

dplyr

```r
# Example
library(dplyr)
library(caret)

# Create new variable
mtcars <- mtcars %>% mutate(power_to_weight = hp / wt)

# Create dummy variables
dummies <- dummyVars(~., data=mtcars)
mtcars_dummies <- predict(dummies, newdata=mtcars)
```

Data processing

- Exploratory data analysis
- Handle missing data
- Outliers
- Feature engineering
- Feature scaling
- Splitting data into training and validation sets

```r
# Example
library(caret)
library(scales)

# Standardize variables
scaled_data <- scale(mtcars)

# Pre-process using caret
preProc <- preProcess(mtcars, method=c("center", "scale"))
mtcars_scaled <- predict(preProc, mtcars)
```

Data processing

- Exploratory data analysis
- Handle missing data
- Outliers
- Feature engineering
- Feature scaling
- Splitting data into training and validation sets

caret

cran/**caTools**

❗ This is a read-only mirror of the CRAN R package repository. caTools — Tools: Moving Window Statistics, GIF, Base64,...

META CRAN

2 Contributors · 0 Issues · 7 Stars · 3 Forks

```r
# Example
library(caret)
library(caTools)

# Using caret
set.seed(123)
trainIndex <- createDataPartition(mtcars$mpg, p=0.7, list=FALSE)
train_data <- mtcars[trainIndex,]
test_data <- mtcars[-trainIndex,]

# Using caTools
set.seed(123)
split <- sample.split(mtcars$mpg, SplitRatio=0.7)
train_data <- subset(mtcars, split==TRUE)
test_data <- subset(mtcars, split==FALSE)
```

Building an ML project

Processing data

Picking and training your ML model

Testing the performance of ML models

Redbubble, Adobe stock

Types of ML algorithms

Supervised learning

- algorithm is trained on a labeled dataset
- each input is paired with the correct output

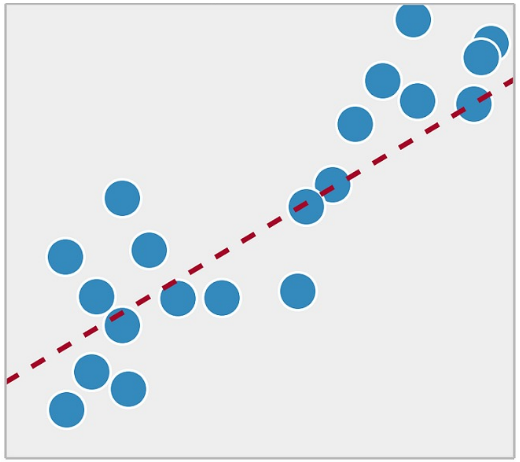
Unsupervised learning

- algorithm works with unlabeled data
- finds patterns or structures
- no explicit guidance

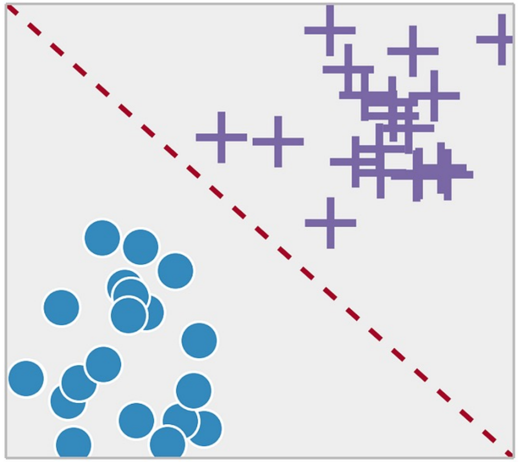
Types of ML algorithms

Supervised learning

Regression

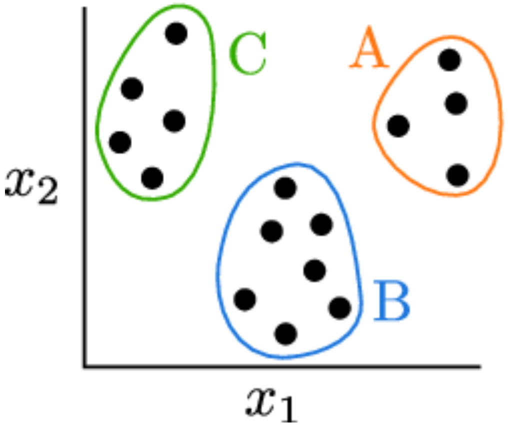


Classification

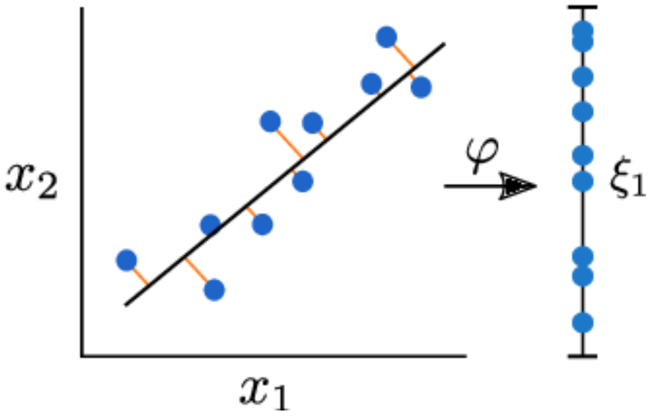


Unsupervised learning

Clustering

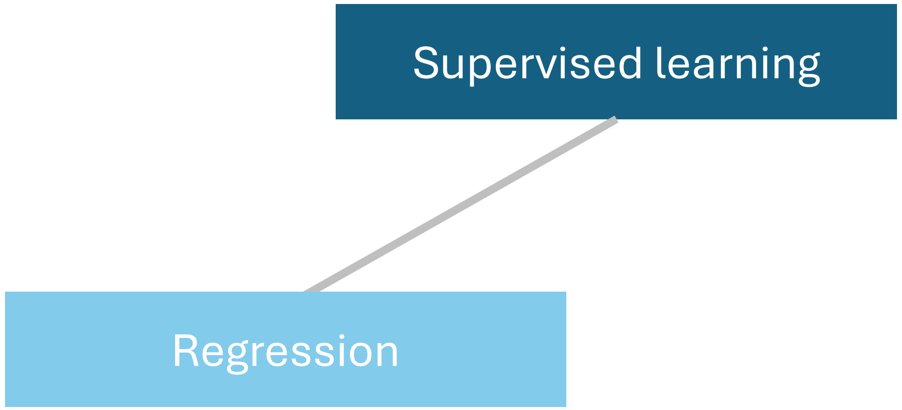


Dimensionality reduction



https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d
https://www.researchgate.net/publication/344783581_A_Perspective_on_Machine_Learning_Methods_in_Turbulence_Modelling

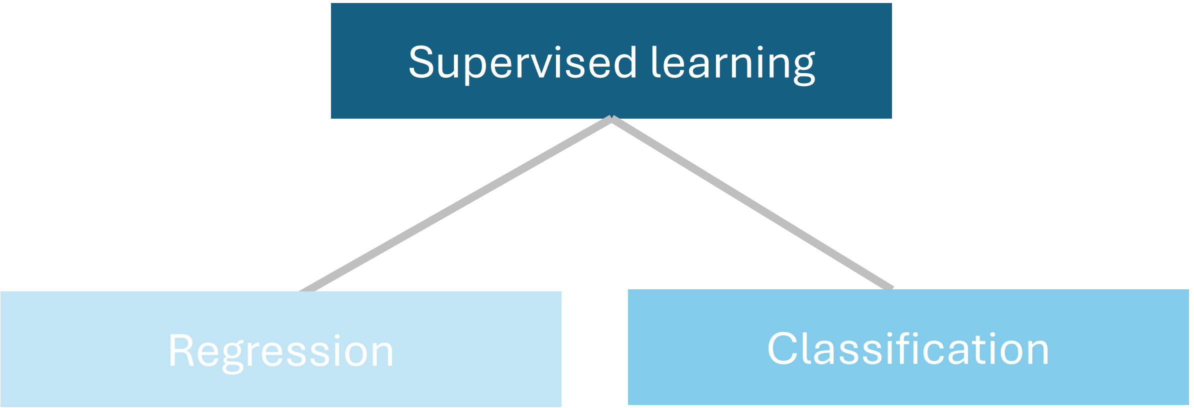
Supervised learning



- **glmnet**: For regularized generalized linear models (lasso, ridge, elastic net)
- **randomForest** or **ranger**: For random forest regression
- **xgboost**: For gradient boosting regression
- **rpart**: For decision tree regression
- **kernlab**: For support vector regression

```
#glmnet: For regularized generalized linear models (lasso, ridge, elastic net)
# Example data
x <- matrix(rnorm(100*20), 100, 20)
y <- rnorm(100)

# Fit a lasso model
fit <- glmnet(x, y, alpha = 1)

#randomForest: For random forest regression
# Example data
data(iris)
set.seed(42)

# Fit a random forest model
fit <- randomForest(Sepal.Length ~ ., data = iris)

#xgboost: For gradient boosting regression
# Example data
data(iris)
set.seed(42)
train <- as.matrix(iris[, -1])
label <- iris$Sepal.Length

# Fit a gradient boosting model
fit <- xgboost(data = train, label = label, nrounds = 100, objective = "reg:squared

#rpart: For decision tree regression
# Example data
data(iris)

# Fit a decision tree model
fit <- rpart(Sepal.Length ~ ., data = iris)

#kernlab: For support vector regression
# Example data
data(iris)

# Fit a support vector regression model
fit <- ksvm(Sepal.Length ~ ., data = iris, type = "eps-svr")
```

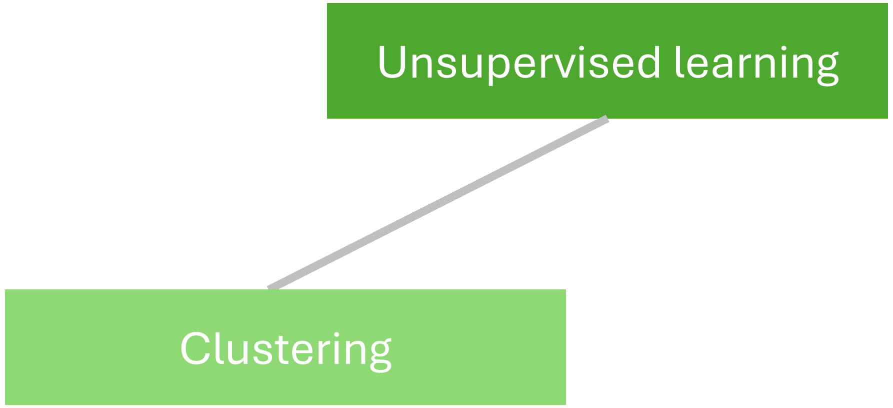
Supervised learning



- **caret:** For various classification algorithms and model training
- **randomForest**: For random forest classification
- **xgboost:** For gradient boosting classification
- **e1071:** For support vector machines and naive Bayes
- **rpart:** For decision tree classification
- **glmnet**: For regularized logistic regression
- **naivebayes:** For naive Bayes classification

```r
#caret: For various classification algorithms and model training
# Example data
data(iris)
set.seed(42)

# Train a model using caret
fit <- train(Species ~ ., data = iris, method = "rf")

#randomForest: For random forest classification
# Example data
data(iris)
set.seed(42)

# Fit a random forest model
fit <- randomForest(Species ~ ., data = iris)

#xgboost: For gradient boosting classification
# Example data
data(iris)
set.seed(42)
train <- as.matrix(iris[, -5])
label <- as.numeric(iris$Species) - 1

# Fit a gradient boosting model
fit <- xgboost(data = train, label = label, nrounds = 100, objective = "multi:softp

#e1071: For support vector machines and naive Bayes
# Example data
data(iris)

# Fit a support vector machine model
fit_svm <- svm(Species ~ ., data = iris)

# Fit a naive Bayes model
fit_nb <- naiveBayes(Species ~ ., data = iris)

#rpart: For decision tree classification
# Example data
data(iris)

# Fit a decision tree model
fit <- rpart(Species ~ ., data = iris)
```

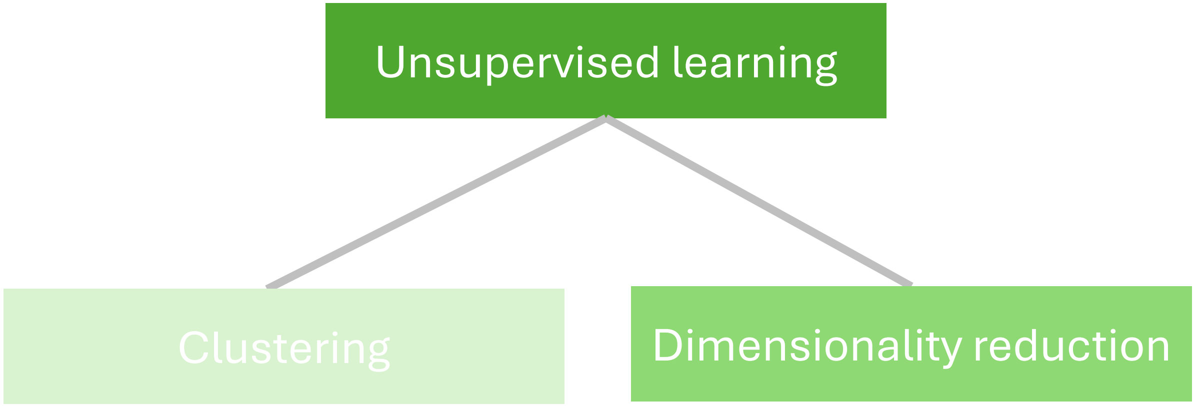
Unsupervised learning



- **stats:** For k-means clustering (built-in R package)
- **cluster:** For various clustering algorithms including hierarchical clustering
- **dbscan:** For density-based clustering

```r
#stats: For k-means clustering (built-in R package)
# Example data
data(iris)

# Perform k-means clustering
fit <- kmeans(iris[, -5], centers = 3)

#cluster: For various clustering algorithms including hierarchical clustering
# Install and load the package
install.packages("cluster")
library(cluster)

# Example data
data(iris)

# Perform hierarchical clustering
fit <- agnes(iris[, -5])

#dbscan: For density-based clustering
# Install and load the package
install.packages("dbscan")
library(dbscan)

# Example data
data(iris)

# Perform DBSCAN clustering
fit <- dbscan(iris[, -5], eps = 0.5, minPts = 5)
```

Unsupervised learning



- **stats**: For principal component analysis (PCA) (built-in R package)
- **FactoMineR:** For various dimensionality reduction techniques
- **Rtsne:** For t-SNE (t-Distributed Stochastic Neighbor Embedding)

```r
#stats: For principal component analysis (PCA) (built-in R package)
# Example data
data(iris)

# Perform PCA
fit <- prcomp(iris[, -5], scale. = TRUE)

#FactoMineR: For various dimensionality reduction techniques
# Install and load the package
install.packages("FactoMineR")
library(FactoMineR)

# Example data
data(iris)

# Perform PCA using FactoMineR
fit <- PCA(iris[, -5], graph = FALSE)

# Rtsne: For t-SNE (t-Distributed Stochastic Neighbor Embedding)
# Install and load the package
install.packages("Rtsne")
library(Rtsne)

# Example data
data(iris)

# Perform t-SNE
fit <- Rtsne(as.matrix(iris[, -5]))
```

Types of ML algorithms

Supervised learning

Regression

glmnet
randomForest or **ranger**
xgboost
rpart
kernlab

Classification

caret
randomForest
xgboost
e1071
rpart
glmnet
naivebayes

Unsupervised learning

Clustering

stats
cluster
dbscan

Dimensionality reduction

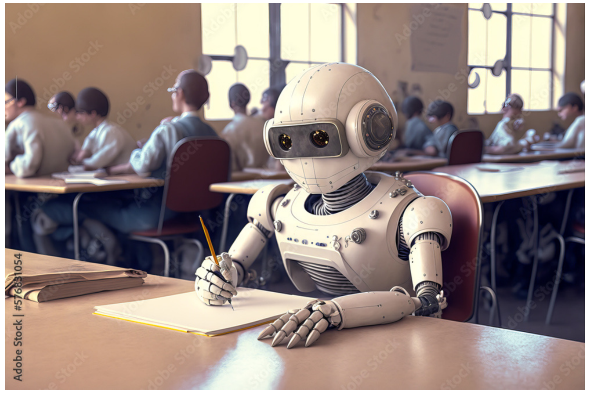
stats or **PCATools**
FactoMineR
Rtsne

Building an ML project

Processing data

Picking and training your ML model

Testing the performance of ML models



Redbubble, Adobe stock

Testing ML models

- **Classification Metrics**
  - Accuracy using **caret**
  - Precision and Recall using **caret**
  - F1 Score using **caret**
  - AUC-ROC using **pROC** or **ROCit**
- **Regression Metrics**
  - RMSE (Root Mean Squared Error)
  - R-squared (R^2)
- **Visualization Techniques**
  - ROC Curves
  - Density Plots and Boxplots

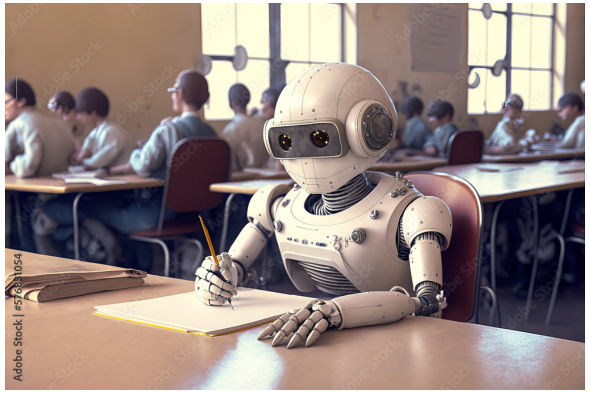
```r
# Load necessary libraries
library(caret)
library(pROC)

# Load dataset
data(mtcars)

# Split data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(mtcars$mpg, p = .8,
                                  list = FALSE,
                                  times = 1)
trainData <- mtcars[ trainIndex,]
testData  <- mtcars[-trainIndex,]

# Make predictions
predictions_prob <- predict(model, testData, type = "response")
predictions <- ifelse(predictions_prob > 0.5, 1, 0)
predictions <- as.factor(predictions)

# Create a confusion matrix
conf_matrix <- confusionMatrix(predictions, testData$Species, positive = "1")
print(conf_matrix)

# Extract metrics
accuracy <- conf_matrix$overall['Accuracy']
precision <- conf_matrix$byClass['Pos Pred Value']
recall <- conf_matrix$byClass['Sensitivity']
f1_score <- 2 * (precision * recall) / (precision + recall)

# ROC Curve
roc_obj <- roc(testData$Species, predictions_prob)
plot(roc_obj)
```

Conclusion

Processing data

Picking and training your ML model

Testing the performance of ML models



Redbubble, Adobe stock

Thank you