Uptrail

STATISTICS & PREDICTIVE MODELLING

WITH PYTHON

Week 3

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

PART 1: BUSINESS CONTEXT & STATISTICAL FOUNDATIONS

PREDICTIVE ANALYTICS IN BUSINESS

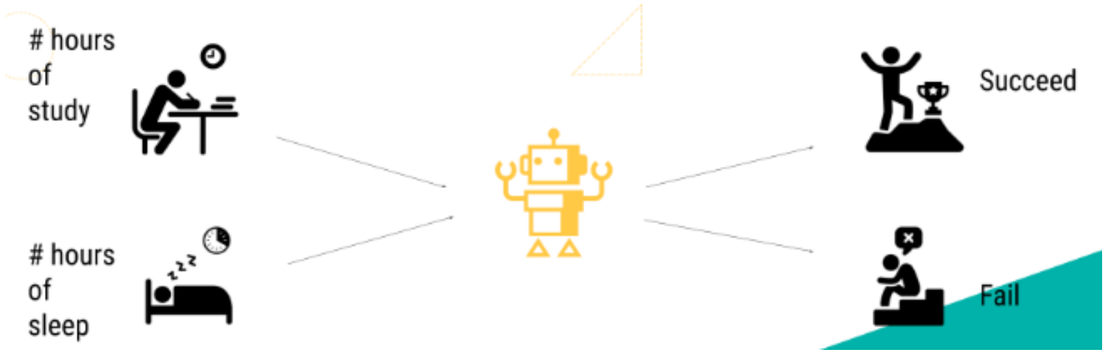
What is Predictive Modelling?

Predictive modelling uses data to forecast future outcomes.

Businesses use it to plan ahead, reduce costs, and improve decision-making.

How to do Predictive Modeling?

Use machine learning/statistics to train models on historical data and predict future outcomes.



Concept | Predictive modeling
https://knowledge.dataiku.com/latest/ml-analytics/ml-concepts/concept-predictive-modeling.html

REAL-WORLD USE CASES

01. Retail

Forecasting next month’s sales

02. Banking

Churn Prediction

Spotting which customers are at risk of leaving (closing their accounts)

03. Logistics

Optimising inventory for seasonal demand

04. SaaS

Campaign Targeting

Choosing the best target audience for ads

CASE STUDY

Scenario:

Imagine you work on the analytics team for Netflix.

“Some customers are watching less... then quietly disappear.”

CustomerID	Age	HoursWatched (per week)	
1001	25	3.5	Yes
1002	45	1.1	No
1003	37	6.7	No
1004	29	0.5	Yes

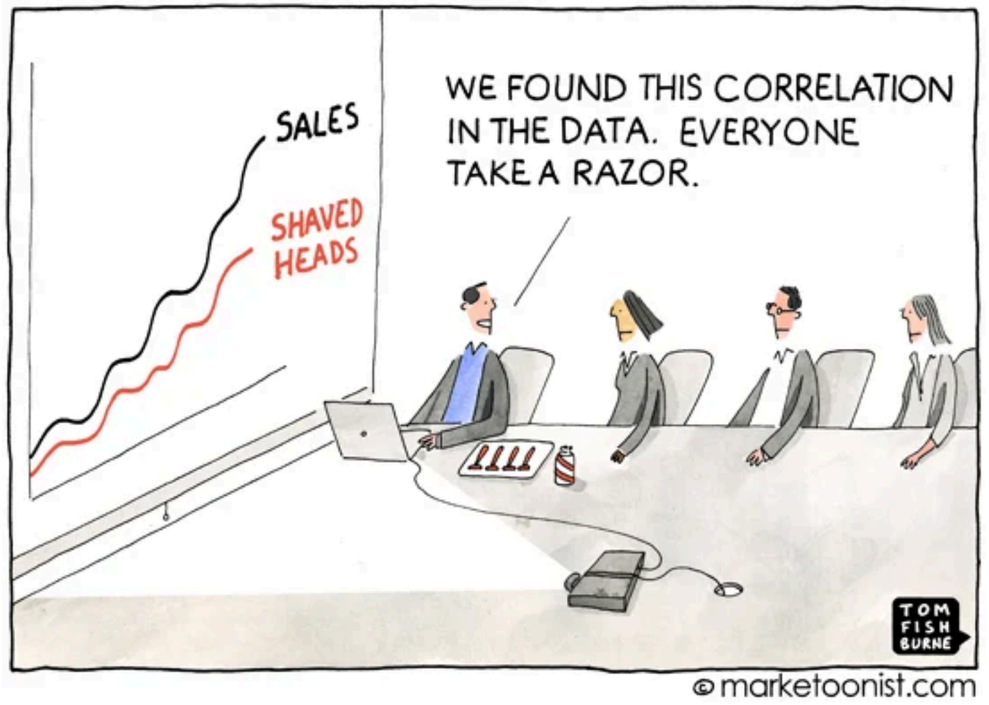
Can we predict who will churn next?

CORRELATION VS. CAUSATION

Correlation: extent to which two or more variables tend to change together.

- positive (both variables increase or decrease together)
- negative (one variable increases as the other decreases)
- zero (no relationship).

Causation: a change in one variable directly causes a change in another variable.



"Just because two things move together doesn't mean one causes the other."

UNDERSTANDING DISTRIBUTIONS

Data distribution helps you identify patterns, trends, and anomalies in datasets. It guides your choice of statistical methods and ensures accurate analysis, leading to better decision-making in real-world applications.

Binomial:

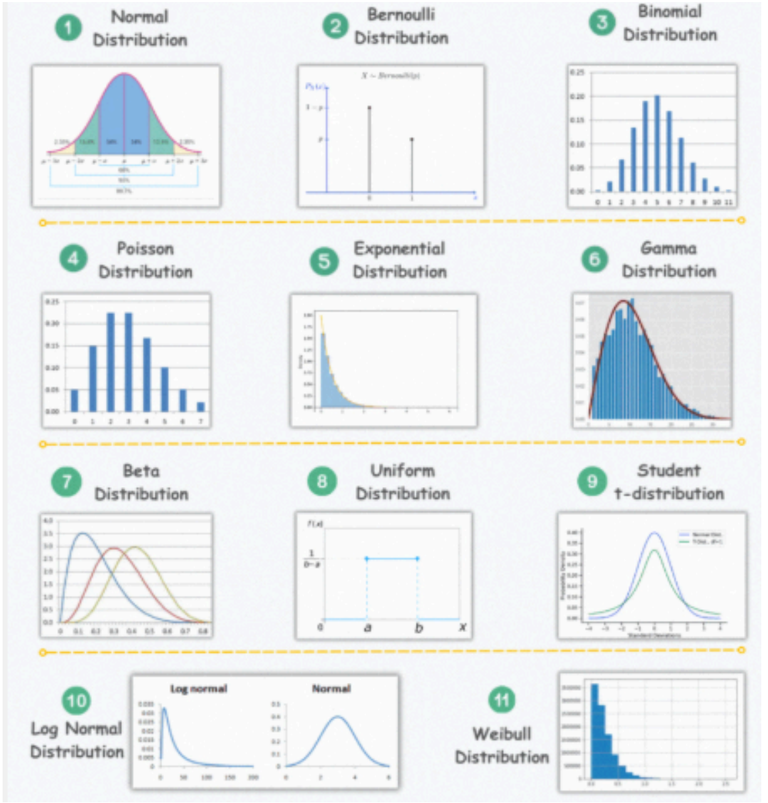
- Models the number of successes in a fixed number of independent trials
- (e.g., number of heads in 10 coin flips).

Poisson:

- Models the number of events occurring within a specific time or location, given an average rate
- (e.g., number of customers arriving at a store in an hour).

Normal (Gaussian):

- A bell-shaped distribution characterized by its mean and standard deviation
- often used to model naturally occurring phenomena.



Statistical Data Distributions
~ https://www.geeksforgeeks.org/engineering-mathematics/introduction-of-statistical-data-distributions/

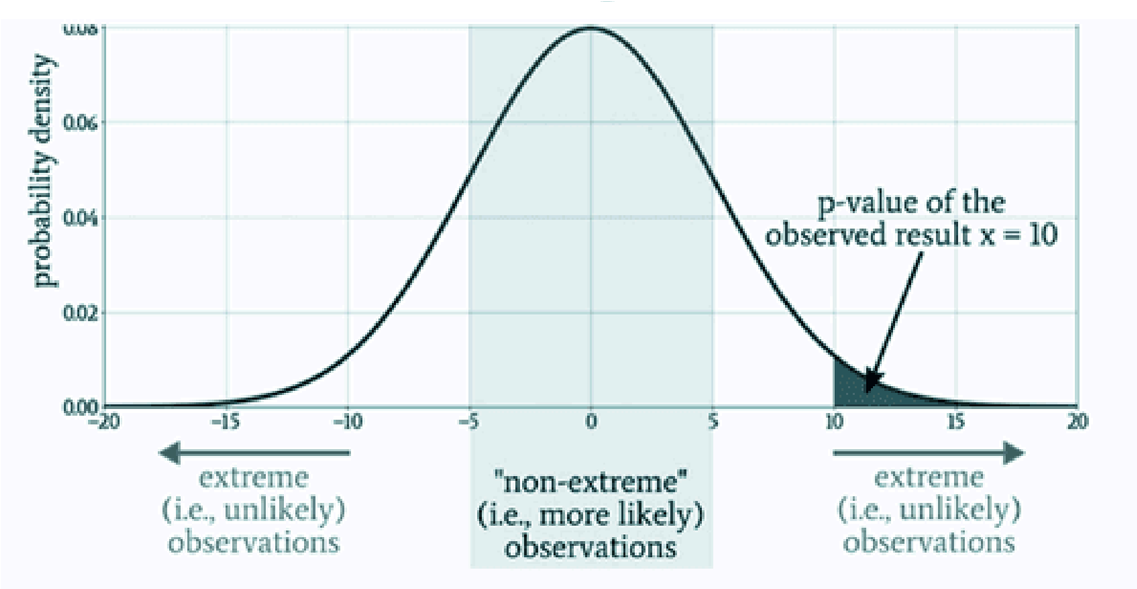
STATISTICAL SIGNIFICANCE & P-VALUES

What is a p-value?

Probability of observing data as extreme as what you have, assuming the null hypothesis is true

Intuition: Small p = stronger evidence against the null

Tip: Statistical significance ≠ business significance



What is P - value | How to Calculate | Statistical Significance
~ https://www.youtube.com/watch?v=jwlGxou0z-M

HYPOTHESIS TESTING

Structured method to test assumptions in data.

Helps validate whether changes or observations are meaningful.

Key Terms:

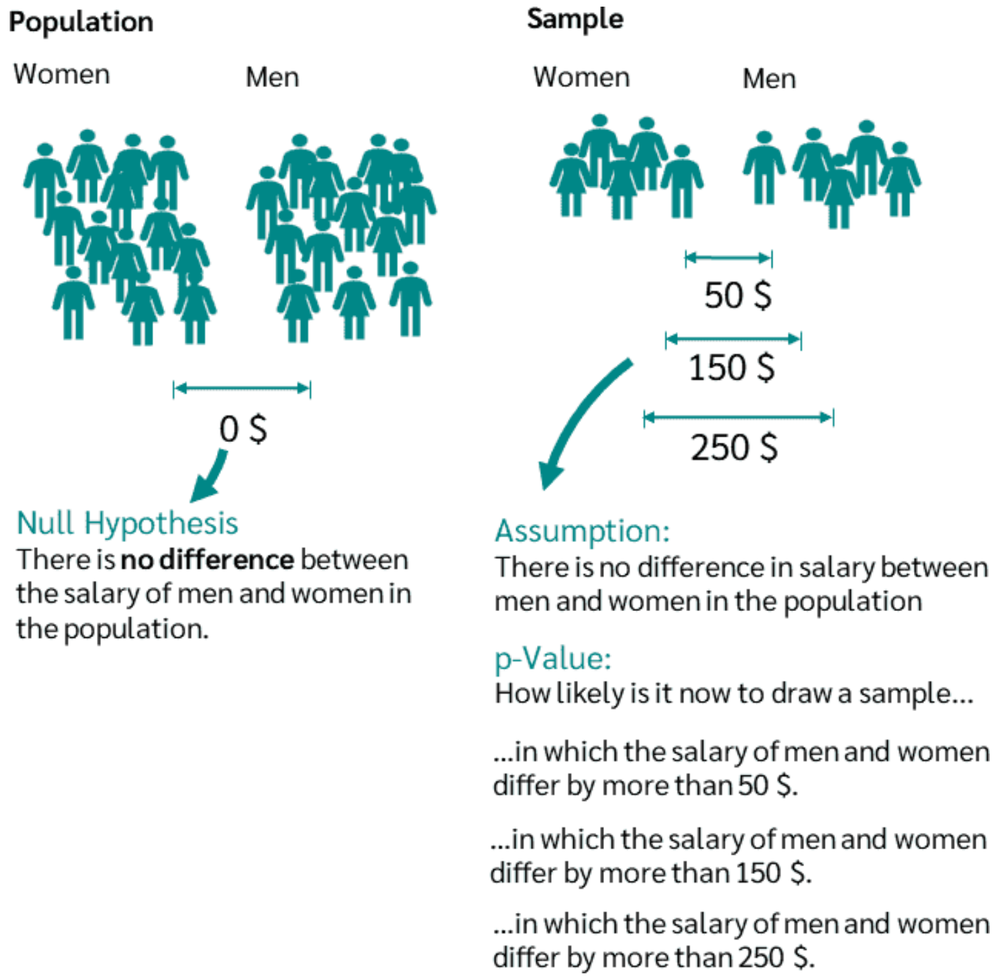
- Null Hypothesis (H_0): no effect
- Alternative Hypothesis (H_1): there is an effect

When to Use Hypothesis Testing?

You want to know if a difference is real (not due to chance).

Conditions to test:

- Comparing outcomes
- Data is collected fairly



A/B TESTING

Method:

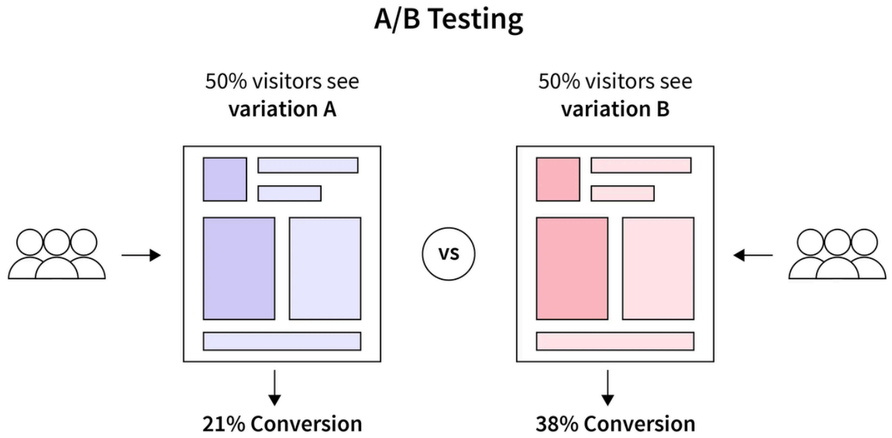
- Two or more versions (A, B, etc.) of the same element are created.
- For example, test different headlines, button colors, or call-to-actions.

Randomization:

- Users are randomly assigned to view one of the variations.

Tracking:

- Track key metrics (like click-through rates, conversion rates, or time spent on page) to measure the performance of each variation.



Mistakes to Avoid:

- Too small sample size
- Running test too short
- Drawing conclusions from random noise

what is A/B testing
~ https://www.youtube.com/watch?v=hHXEjcJkcbc&t=65

A/B TESTING

Method:

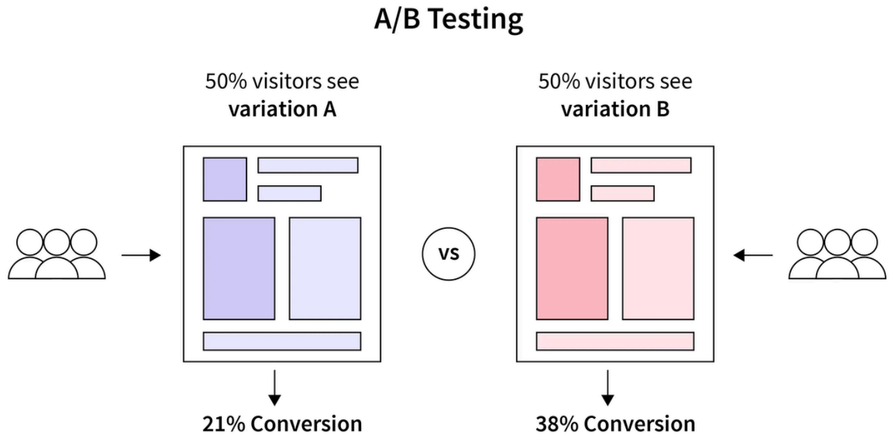
- Two or more versions (A, B, etc.) of the same element are created.
- For example, test different headlines, button colors, or call-to-actions.

Randomization:

- Users are randomly assigned to view one of the variations.

Tracking:

- Track key metrics (like click-through rates, conversion rates, or time spent on page) to measure the performance of each variation.



Mistakes to Avoid:

- Too small sample size
- Running test too short
- Drawing conclusions from random noise

what is A/B testing
~ https://www.youtube.com/watch?v=hHXEjcJkcbc&t=65

PART 2: PREDICTIVE MODELLING WITH SCIKIT-LEARN

WHAT IS SCIKIT-LEARN?

Scikit-learn: Machine Learning Toolbox in Python

- Most popular Machine Learning library in Python
- Simple, consistent API across all algorithms
- Built on NumPy, SciPy, Matplotlib
- 50+ algorithms, preprocessing tools, model evaluation

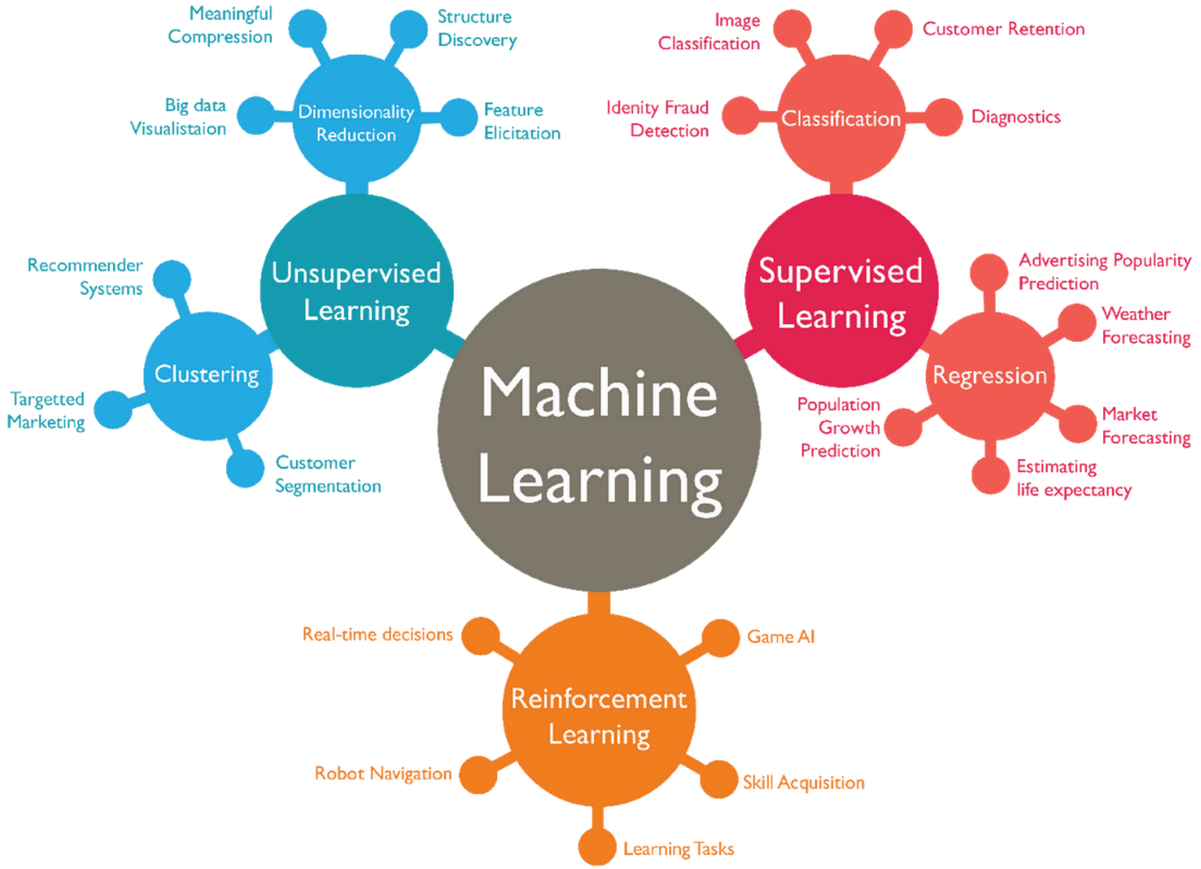
WHAT IS MACHINE LEARNING (ML)?

ML is about **teaching computers to learn from data**

It finds patterns and makes predictions **without being explicitly programmed**

Two main types we'll cover:

- **Regression:** Predicting numbers (e.g. revenue)
- **Classification:** Predicting categories (e.g. churn vs no churn)



THE UNIVERSAL ML WORKFLOW

Every ML Project Follows This Pattern

1

Load & Explore Data:
Understand what you're working with

2

Split Data (Train/Test):
Keep some data hidden for testing

3

Preprocess Features:
Clean and prepare data for algorithms

4

Train Model:
Let the algorithm learn patterns

5

Make Predictions:
Use the trained model on new data

6

Evaluate Performance:
Measure how well it works

CORE FUNCTIONS

Your Scikit-learn Toolkit *(these are building blocks repeated across all algorithms.)*

- **train_test_split():** split data
- **fit():** train model
- **predict():** make predictions
- **StandardScaler:** feature scaling
- **LabelEncoder:** encoding categorical data

```python
# Data splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Preprocessing
from sklearn.preprocessing import StandardScaler, LabelEncoder
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Model training (same pattern for ALL algorithms)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

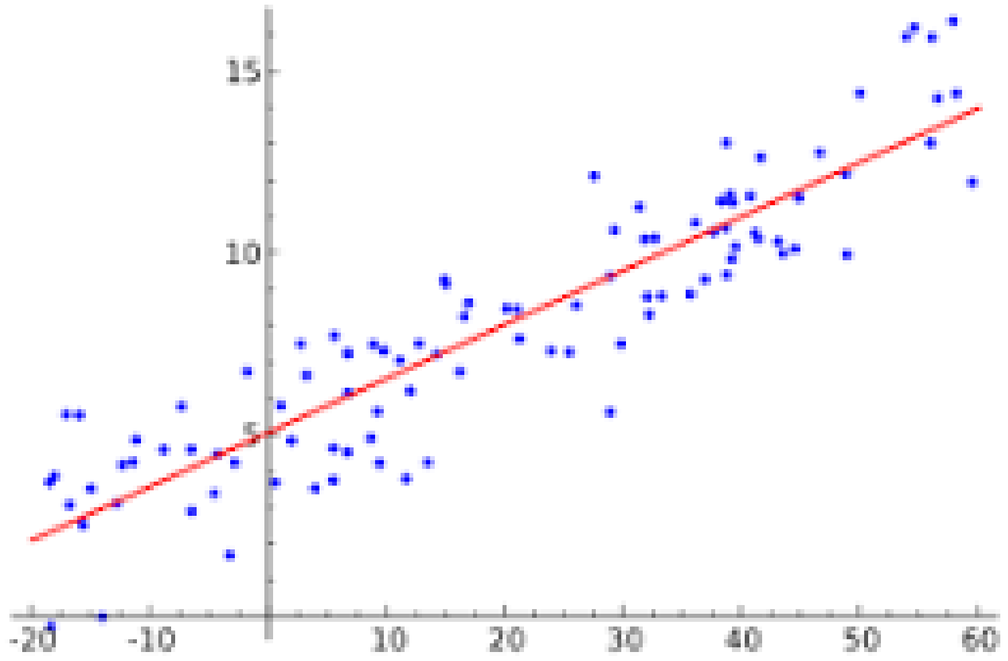
* **Scaling is covered in week 2**

LINEAR REGRESSION: Finding the Best Line

Predicts continuous values (prices, sales, quantities)

Finds best straight line through data points

Formula: y = mx + b (slope × input + intercept)



KEY CONCEPTS of Linear Regression

- **Dependent Variable (Y):** The variable being predicted or explained.
- **Independent Variable(s) (X):** The variable(s) used to predict the dependent variable.
- **Linear Equation:** The relationship between the variables is assumed to be linear, meaning it can be represented by a straight line (or a plane in higher dimensions).
- **Coefficients:** The parameters of the linear equation that determine the line's slope and intercept (and their higher-dimensional counterparts).
- **Residuals:** The differences between the observed values and the predicted values.
- **Best-fit Line:** The line that minimizes the sum of the squared residuals (least squares method).

USE CASE PREVIEW

- **Features:** ad_spend, website_traffic, email_campaigns, social_media_reach
- **Target:** monthly_sales (regression) and high_value_customer (classification)
- **Size:** 1,000 customers, 4 features
- **Business Goal:** Optimize marketing spend

We will Predict:

- **Linear Regression:** Predict monthly sales from marketing spend
- **Logistic Regression:** Predict if customer will churn

E-commerce Marketing Dataset

	ad_spend	website_traffic	email_campaigns	social_media_reach	monthly_sales	high_value_customer
0	2397.37	8591	9	26044	16224.78	0
1	1889.39	6593	8	22012	11228.20	0
2	2518.15	7614	10	39666	14742.75	0
3	3218.42	9008	4	51249	20092.86	0
4	1812.68	6136	3	15298	10155.53	0

BUSINESS SCENARIO for Linear Regression

Predicting Revenue from Ad Spend

- **Question:** "If I spend £1,000, what revenue can I expect?"
- **Input:** Marketing spend across channels
- **Output:** Expected monthly revenue
- **Feature:** ad_spend
- **Target:** revenue
- **Business Value:** ROI calculation, budget planning

IMPLEMENTATION of Linear Regression

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Prepare data for linear regression
X = df[['ad_spend']]
y = df['monthly_sales']

# Split data into 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 1: Create model
lr_model = LinearRegression()

# Step 2: Train model
lr_model.fit(X_train, y_train)

# Step 3: Make predictions
y_pred = lr_model.predict(X_test)

# Step 4: Visualize results
plt.scatter(X_test, y_test, alpha=0.5)
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.title('Revenue vs Ad Spend')
plt.xlabel('Ad Spend (£)')
plt.ylabel('Monthly Revenue (£)')
```

MODEL INTERPRETATION for Linear Regression

What the Model Tells Us?

- **Interpretation:**
- **Coefficient (5.14):** Every £1 spend in ads has effect of £5.14 in revenue
- **Intercept (1656.90):** Base revenue with £0 ad spend
- **R² (0.75):** Model explains 75% of variance
- **RMSE (2227.23):** Model's predictions are, on average, off by £2,227.

```
Linear Regression Results:
Coefficient    : 5.14
Intercept      : £ 1656.90
R² Score       : 0.755
RMSE           : £ 2227.23
```

```python
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Evaluate
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"Linear Regression Results:")
print(f"{'Coefficient':15s}: {lr_model.coef_[0]:.2f}")
print(f"{'Intercept':15s}: £{lr_model.intercept_:8.2f}")
print(f"{'R² Score':15s}: {r2:.3f}")
print(f"{'RMSE':15s}: £ {rmse:.2f}")
```

MULTIPLE FEATURES

Real Models Use Multiple Inputs for Regression

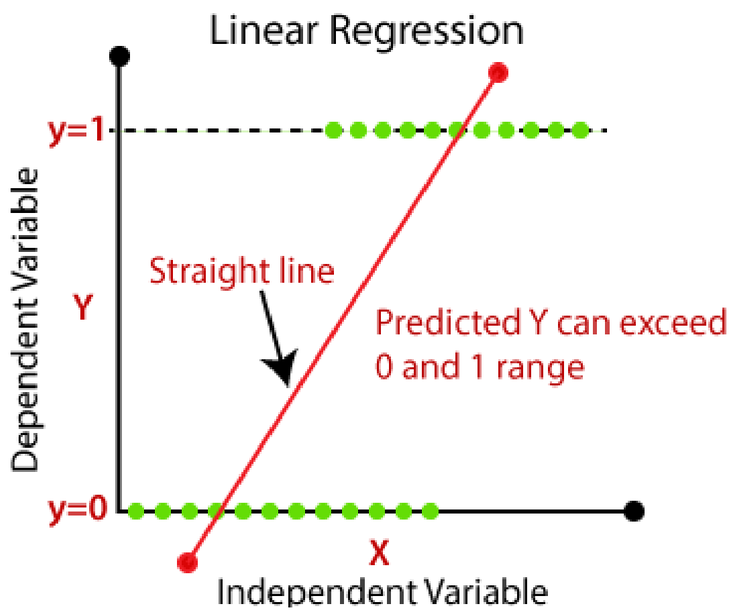
- **Interpretation:**
- **Intercept (92.69):** Base revenue when all predictors are £0
- **R² (0.81):** 81% of the variability in the revenue is explained by all the independent variables.
- **RMSE (1941.28):** Model's predictions are, on average, £1,941 farther from the actual values.
- **ad_spend (2.69):** Every £1 spend in ads has effect of £2.69 in revenue [and similarly for other coeeficients]

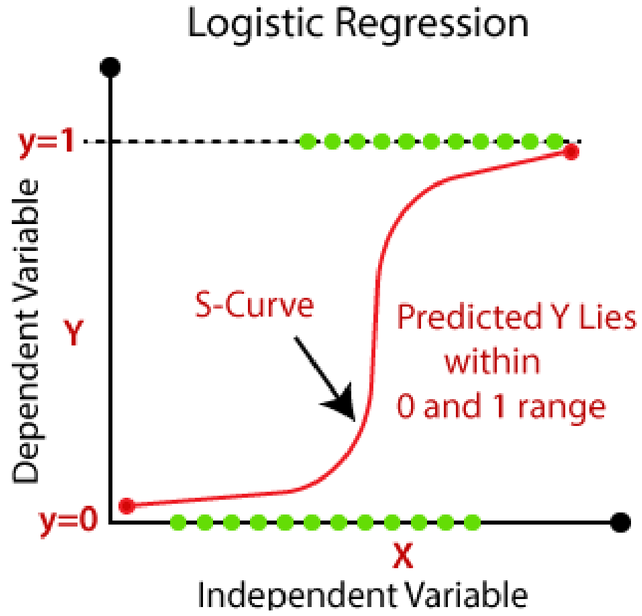
```
Linear Regression Results:
R² Score: 0.814
RMSE: £ 1941.28
Intercept: £ 92.69

Feature Coefficients:
ad_spend            : £    2.69
website_traffic     : £    0.81
email_campaigns     : £  131.30
social_media_reach  : £    0.01
```

```python
# Prepare data for linear regression
features = ['ad_spend', 'website_traffic', 'email_campaigns', 'social_media_reach']
X = df[features]
y = df['monthly_sales']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 1: Create model
lr_model = LinearRegression()

# Step 2: Train model
lr_model.fit(X_train, y_train)

# Step 3: Make predictions
y_pred = lr_model.predict(X_test)

# Evaluate
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"Linear Regression Results:")
print(f"R² Score: {r2:.3f}")
print(f"RMSE: £ {rmse:.2f}")
print(f"Intercept: £ {lr_model.intercept_:.2f}")
print()
print("Feature Coefficients:")
for feature, coef in zip(features, lr_model.coef_):
    print(f"{feature:20s}: £{coef:8.2f}")
```

LOGISTIC REGRESSION: For Yes/No Questions

- Predicts probabilities (0 to 1)
- Uses sigmoid function (S-shaped curve)
- Perfect for binary classification

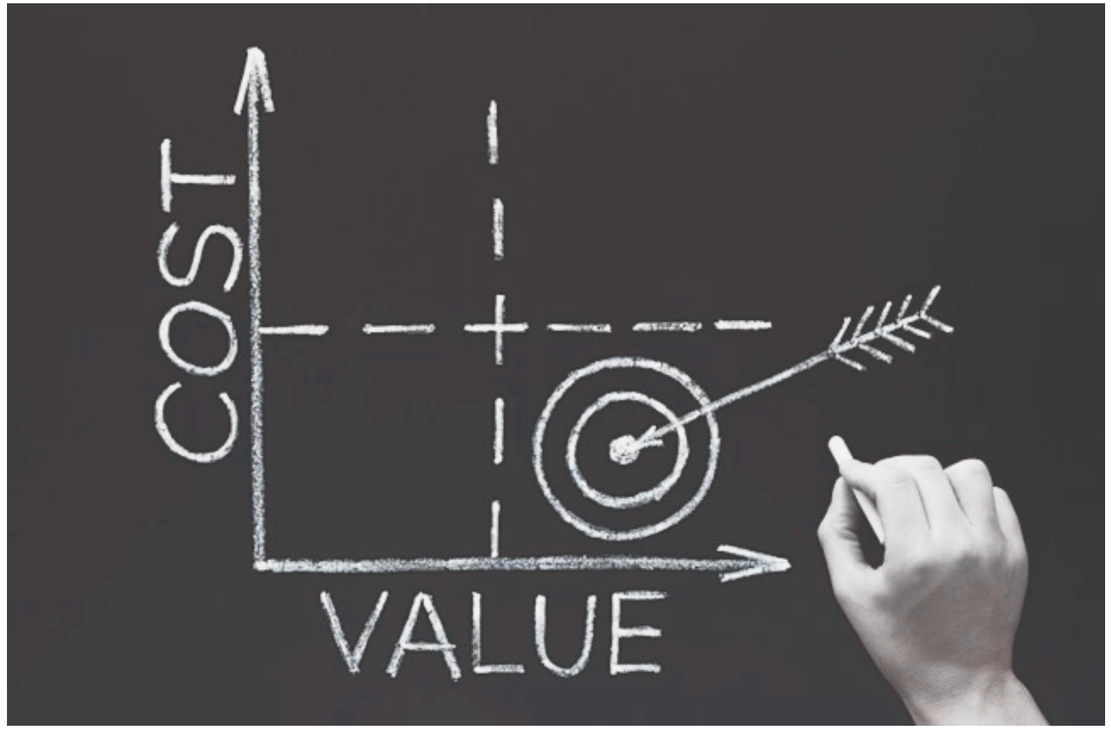




BUSINESS SCENARIO for Logistic Regression

Predicting High-Value Customers

- **Question:** "Will this customer be a high-value customer (>$25K sales)?"
- **Input:** Marketing spend, website traffic, email campaigns, social media reach
- **Output:** Probability of being high-value + yes/no prediction
- **Business Value:** Targeted marketing, resource allocation, VIP treatment



IMPLEMENTATION of Logistic Regression

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Prepare data for logistic regression
X = df[['monthly_sales']]
y = df['high_value_customer']

# Split data into 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 1: Create model
log_model = LogisticRegression(random_state=42)

# Step 2: Train model
log_model.fit(X_train, y_train)

# Step 3: Make predictions
y_pred = log_model.predict(X_test)
y_pred_proba = log_model.predict_proba(X_test)[:, 1]

# Step 4: Create realistic test cases based on actual patterns
test_customers = [
    # Low-value profile (below average in most metrics)
    [18000],
    # Medium-value profile (around average)
    [25000],
    # High-value profile (above average in most metrics)
    [35000],
    # Very high-value profile (top tier across all metrics)
    [45000]
]

print("\nPredictions on realistic customer profiles:")
for i, customer in enumerate(test_customers):
    pred = log_model.predict([customer])
    pred_proba = log_model.predict_proba([customer])
    profile = ['Low-Value', 'Medium-Value', 'High-Value', 'Very High-Value'][i]
    print(f"{profile} Profile: {'High-Value' if pred[0] == 1 else 'Regular'} "
          f"(Probability: {pred_proba[0][1]:.3f})")
    print(f"  [Sales: £{customer[0]}]")
```

```
Predictions on realistic customer profiles:
Low-Value Profile: Regular (Probability: 0.165)
  [Sales: £18000]
Medium-Value Profile: Regular (Probability: 0.416)
  [Sales: £25000]
High-Value Profile: High-Value (Probability: 0.816)
  [Sales: £35000]
Very High-Value Profile: High-Value (Probability: 0.965)
  [Sales: £45000]
```

Understanding Probabilities:

- customer_1: 0.16 probability → Prediction: Not High Value (0)
- customer_2: 0.41 probability → Prediction: Not High Value (0)
- customer_3: 0.81 probability → Prediction: Will High Value (1)
- customer_4: 0.96 probability → Prediction: Will High Value (1)

Key Point: Default threshold = 0.5, but you can adjust this!

- High threshold: Focus marketing budget on sure high-value customers
- Low threshold: Broader VIP treatment, risk of wasted resources

MODEL EVALUATION METRICS

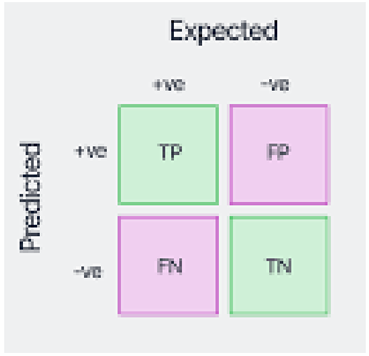
The Accuracy Trap

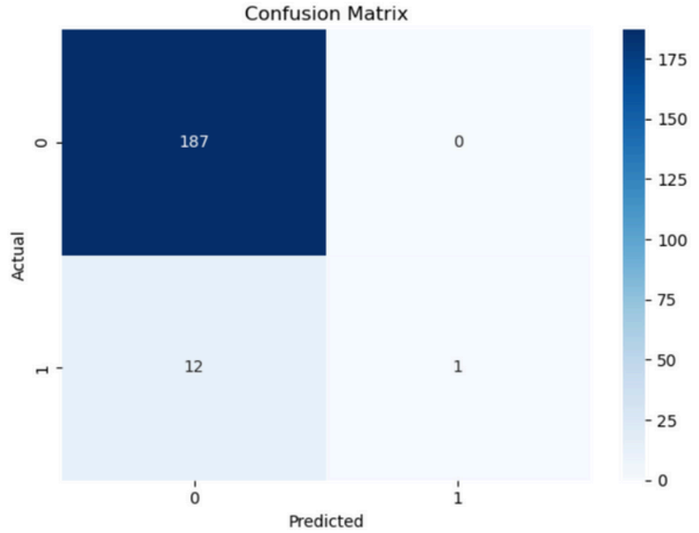
- **Scenario:** High Value Customer classifier with 95% accuracy
- Sounds great, right?
- But what if only 5% of customers are high value?
- A model that says "not High Value" for everything gets 95% accuracy!

- **Key Point: Need multiple metrics for complete picture**

CONFUSION MATRIX: Truth Table

Matrix help in understanding how often a model correctly classifies data points and where it makes errors.





- The cells of the matrix show the number of instances that fall into each category
- **True Positive (TP):** Correctly predicted positive cases.
- **True Negative (TN):** Correctly predicted negative cases.
- **False Positive (FP):** Incorrectly predicted as positive (Type I error).
- **False Negative (FN):** Incorrectly predicted as negative (Type II error).

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.tight_layout()
plt.show()
```

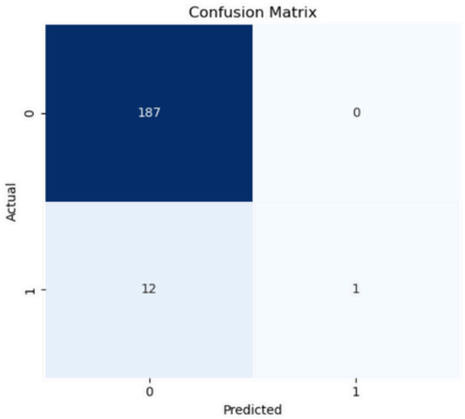
THE FOUR METRICS:

Accuracy, Precision, Recall, F1 Score

Definitions:

- **Accuracy:** (TP + TN) / Total → Overall correctness across all predictions
- **Precision:** TP / (TP + FP) → "Of my positive predictions, how many were right?"
- **Recall: TP / (TP + FN)** → "Of all actual positives, how many did I find?"
- **F1: Harmonic mean** → Balance between precision and recall

```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       187
           1       1.00      0.08      0.14        13

    accuracy                           0.94       200
   macro avg       0.97      0.54      0.56       200
weighted avg       0.94      0.94      0.92       200
```




```python
# Step 5: Evaluate model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.3f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

WHICH METRIC MATTERS MOST?

Business Examples

High Precision Needed:

- VIP program enrollment → Don't waste resources on wrong customers
- Premium product recommendations → Avoid irrelevant suggestions

High Recall Needed:

- High-value customer identification → Don't miss potential big spenders
- Loyalty program invitations → Cast wide net for retention

Balanced F1 Needed:

- Marketing budget allocation → Balance targeting accuracy with coverage

RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

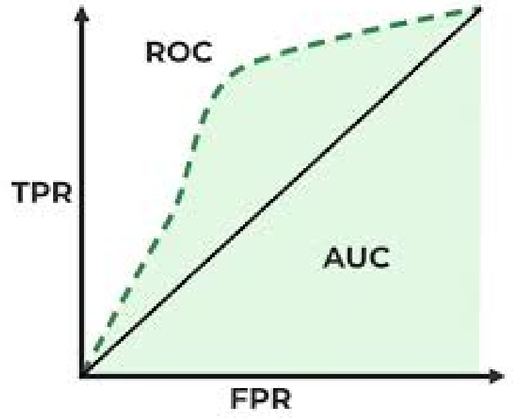
Shows model performance across all thresholds

X-axis: False Positive Rate (FPR) or 1-Specificity:

- The proportion of actual negatives that are incorrectly identified as positive.

Y-axis: True Positive Rate (TPR) or Sensitivity:

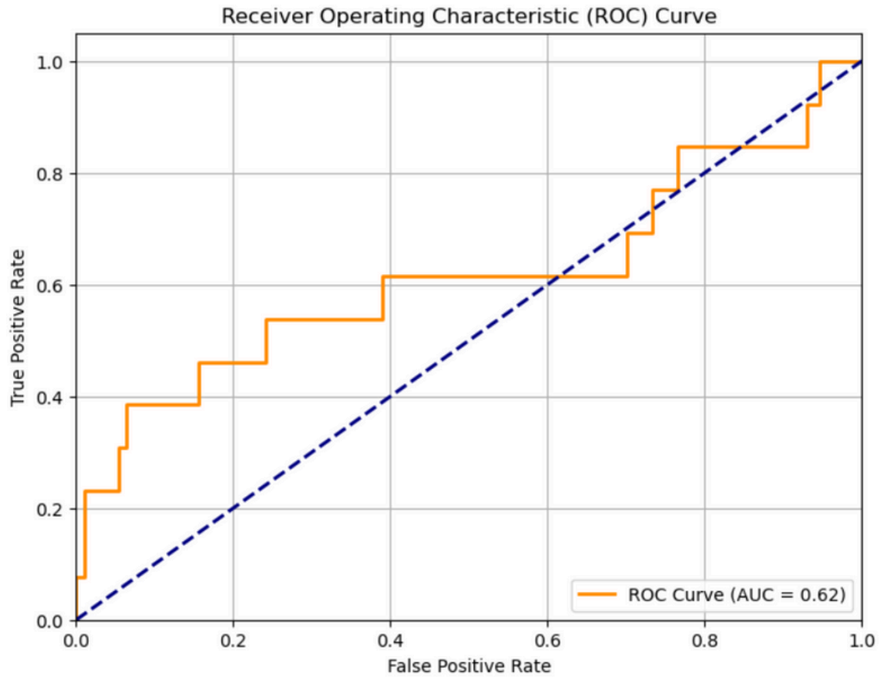
- The proportion of actual positives that are correctly identified as positive.



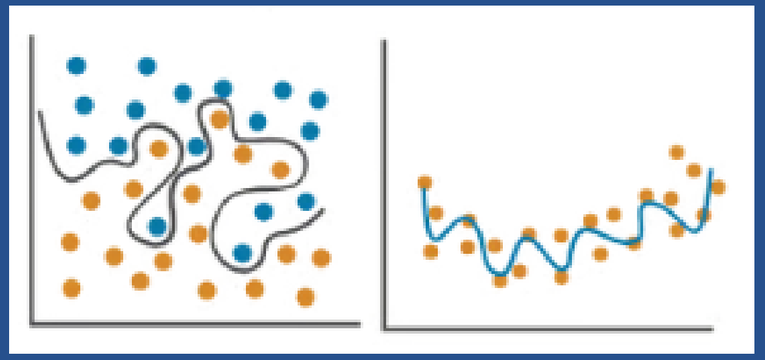
- **Interpretation:**
- A **good classifier** will have a curve that bows towards the top-left corner, indicating **high sensitivity and low FPR.**
- A **diagonal line** from the bottom-left to the top-right represents a **random classifier** (no discriminatory power).
- The **area under the ROC curve (AUC)** provides a single scalar value representing the classifier's overall **performance**. An AUC of 1 indicates perfect classification, while 0.5 indicates a random classifier.

PLOTTING ROC CURVE

```python
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Assuming you already have:
# y_test = true labels
# y_prob = predicted probabilities from model (not class labels)

# 1. Get the False Positive Rate (fpr), True Positive Rate (tpr), and thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

# 2. Calculate the Area Under the Curve (AUC)
roc_auc = auc(fpr, tpr)

# 3. Plot the ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')  # Diagonal reference line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



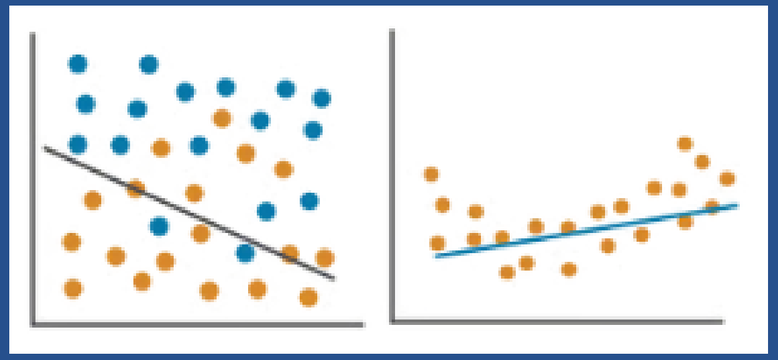
ILLUSION OF 100% ACCURACY



Overfit

A model that learns both the patterns and the noise in the training data.

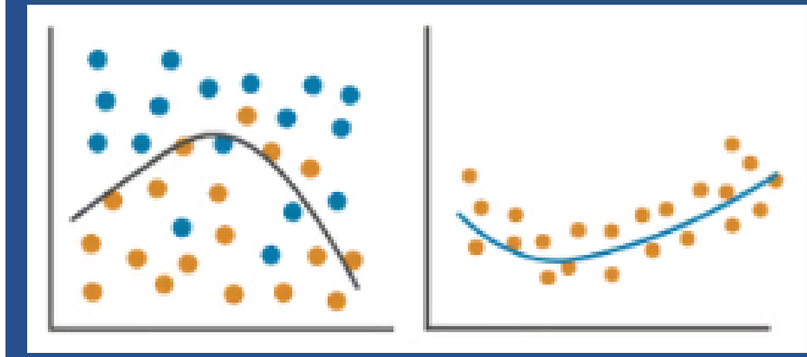
- high variance
- Excellent performance on training data
- Poor generalisation to new/unseen data



Underfit

A model that is too simple to capture the underlying structure of the data.

- High bias, low variance
- Poor performance on both training and test data
- Fails to learn the patterns



Optimizedfit

A balanced model that generalises well to unseen data.

- Low bias, low variance
- Good performance on both training and test sets
- Captures the true structure without overcomplicating

MODEL TUNING

Why Tune?

(Default Settings Aren’t Always Right)

- Every algorithm has "knobs" to adjust → Hyperparameters
- Hyperparameters affect training
- Default values work okay, rarely optimal
- Small tweaks → big impact

(Like adjusting camera settings for different lighting)

Methods of Model Tunning
https://www.anyscale.com/blog/what-is-hyperparameter-tuning

SURVIVAL ANALYSIS

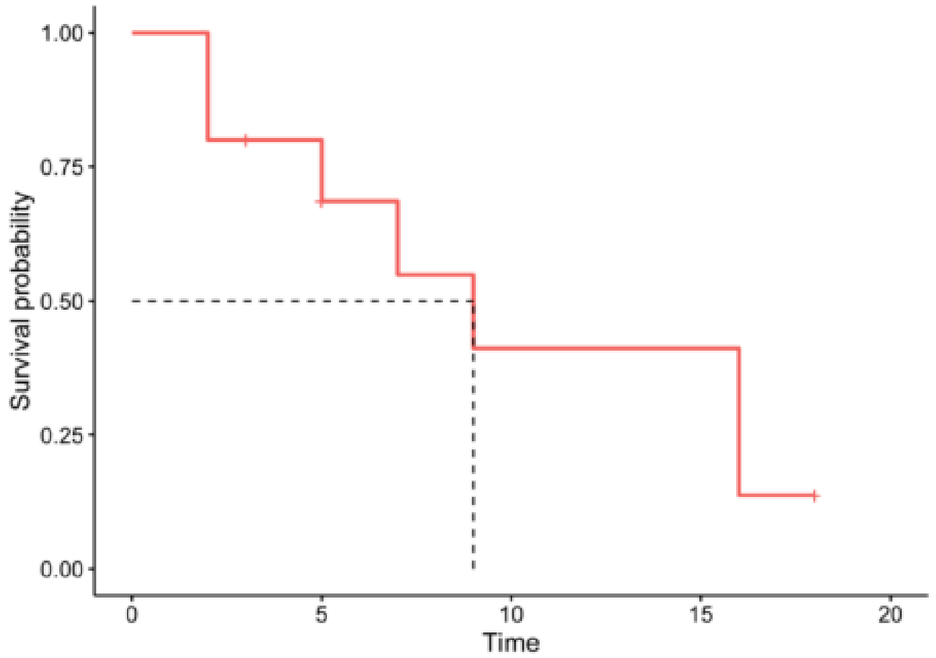
A set of statistical methods for analysing **time-to-event** data, such as time until failure, recovery, or death.

Key Concepts:

- **Censoring:** Event has not occurred during the observation period.
- **Survival Function (S(t)):** Probability of surviving beyond time t.
- **Hazard Function:** Instantaneous risk of the event occurring.

Common Models:

- Kaplan-Meier Estimator
- Cox Proportional Hazards Model



TIME SERIES FORECASTING

Predicting future values based on previously observed data points over time.

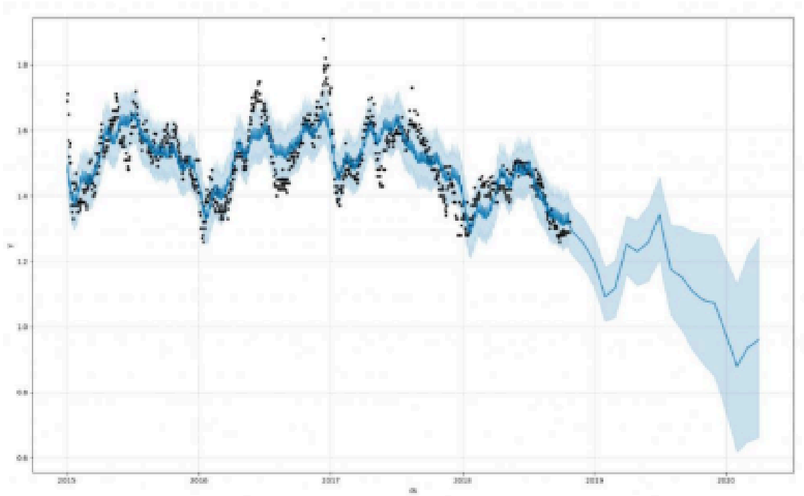
- Data is time-dependent and sequential.
- Seasonality, trend, and noise may be present.

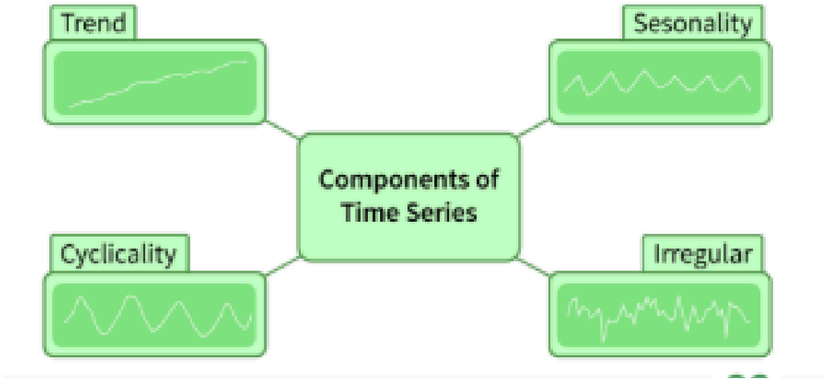
Common Techniques:

- ARIMA / SARIMA
- Exponential Smoothing
- LSTM (Deep Learning)
- Prophet (by Facebook)

Applications:

Stock market, sales prediction, weather forecasting





ENSEMBLE LEARNING

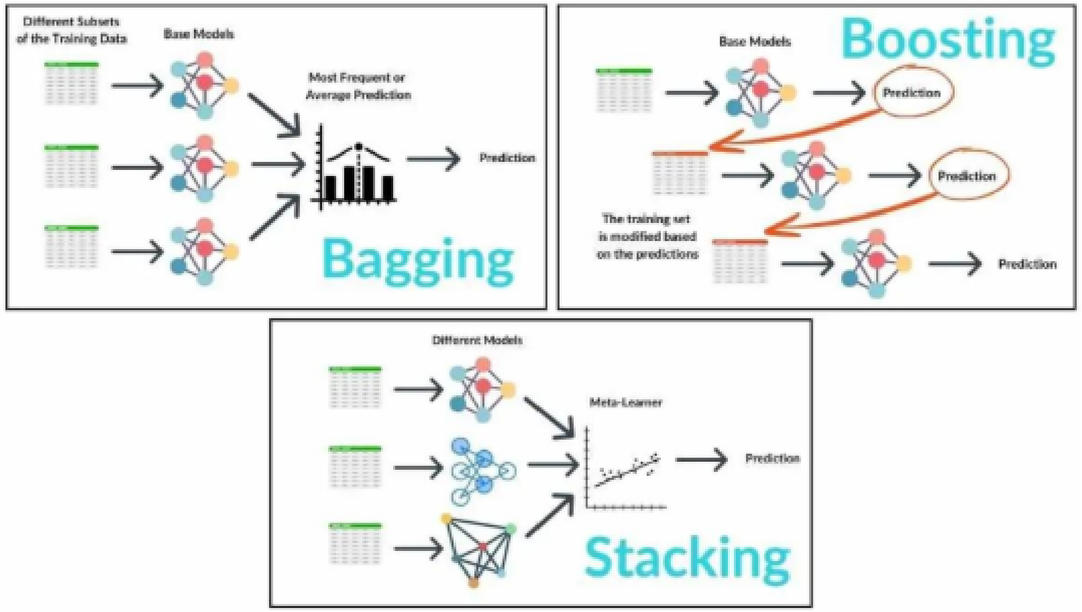
Combining multiple models to produce a stronger overall model.

Types:

- Bagging: Parallel combination (e.g., Random Forest)
- Boosting: Sequential combination (e.g., XGBoost, AdaBoost)
- Stacking: Models combined by a meta-model

Advantages:

- Reduces variance and bias
- Improves accuracy and robustness



SUPPORT VECTOR MACHINES (SVM)

A supervised learning algorithm that finds the optimal hyperplane to separate classes.

Key Concepts:

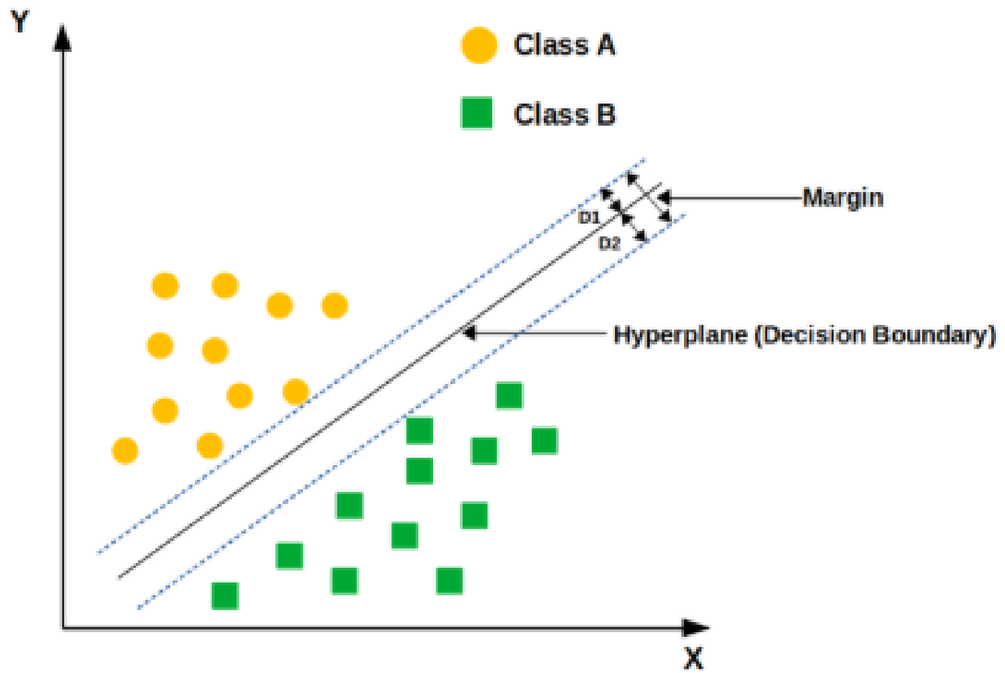
- Maximises margin between classes
- Can handle linear and non-linear data using kernels

Common Kernels:

- Linear
- Polynomial
- Radial Basis Function (RBF)

Use Cases:

Text categorisation, image classification, bioinformatics



NEURAL NETWORKS

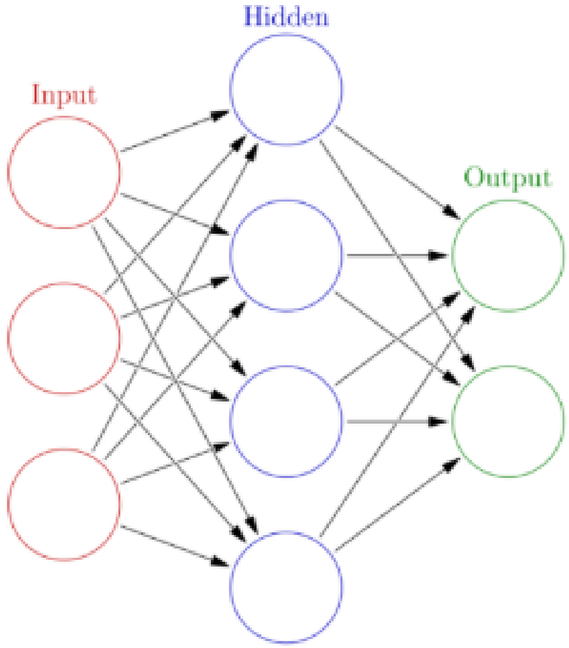
A computational model inspired by the human brain, consisting of layers of interconnected neurons.

Variants:

- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Transformer Models

Applications:

- Image recognition, speech processing, language translation



NATURAL LANGUAGE PROCESSING (NLP) AND TEXT ANALYTICS

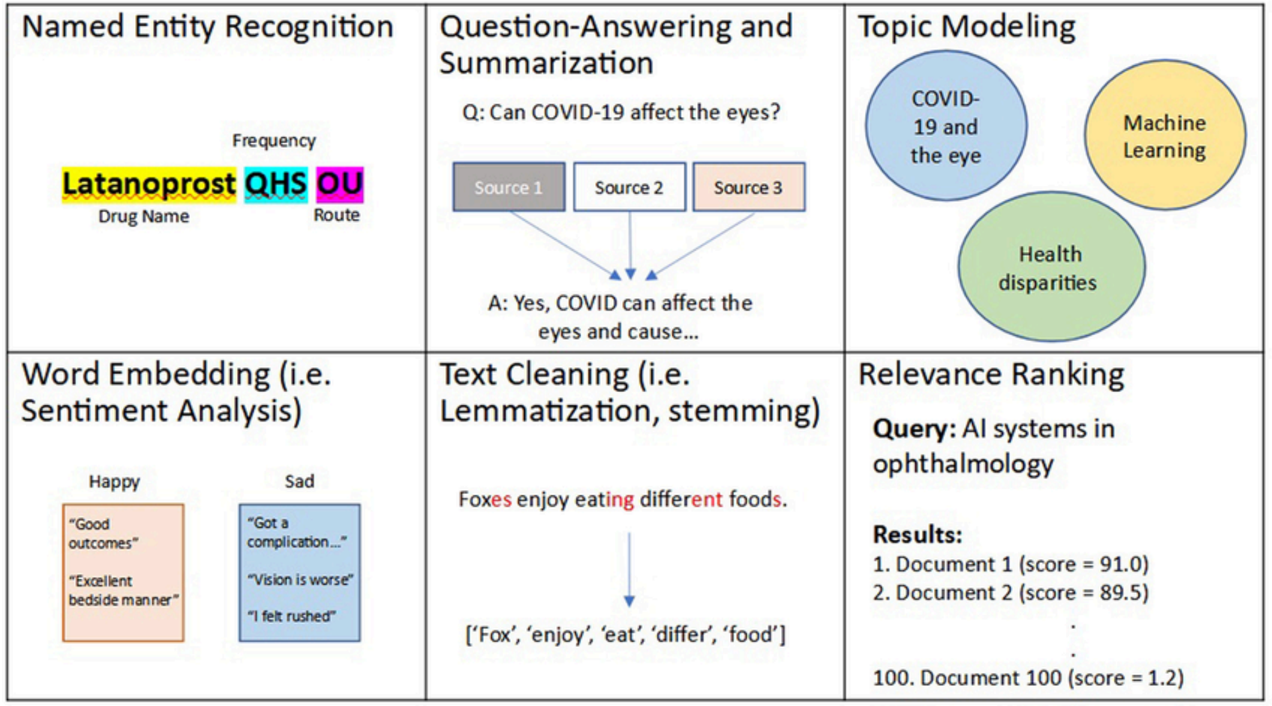
Techniques to enable machines to understand, interpret, and generate human language.

Common Tasks:

- Tokenisation, stemming, lemmatisation
- Named Entity Recognition (NER)
- Sentiment Analysis
- Text Classification and Clustering

Tools & Libraries:

- NLTK, SpaCy, Hugging Face Transformers
- Word2Vec, BERT, GPT models



Applications:

Chatbots, document summarisation, translation, social media analysis

Uptrail

QUESTIONS?

www.uptrail.co.uk

support@uptrail.com