

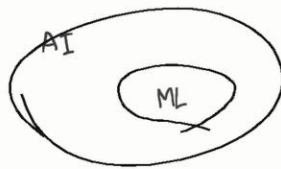
week 1

ML

- enables computers to learn without being explicitly programmed (Arthur Samuel 1959)
- Tom Mitchell = a program learns from experience (E) regarding a task (T) & a performance measure (P) if its performance improves w that experience
 - eg weather forecasting app experience = historical weather data
 - task = predicting today's weather perf = rains or not
- real world = face recognition spam filters in email recommendations (Netflix) fraud detection (Paypal) gen AI (GPT)

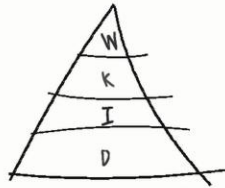
AI vs ML

- AI (broader) → creating a program that can sense act & adapt to mimic human behaviour
 - decision making
 - simulating natural intelligence to solve complex problems → leads to intelligence + wisdom
- ML = subset of AI
 - technique designed to make a machine improve its performance on a specific task by learning from data & experience
 - creating self-learning algorithms → knowledge
 - chess program w hand-written rules = AI but not ML (nothing is learned)
 - spam filter trained on my mails = ML



data vs information

- data = raw unprocessed facts numbers text or symbols
- lacking context no meaning on its own
- information = data all processed or contextualized to make it meaningful & useful
 - answers who what where when
 - DIKW pyramid (Data Information Knowledge Wisdom) = how raw facts are progressively transformed into actionable human judgement



- D = discrete objective facts or observations
- most abundant least valued (database of timestamps & product codes)
- I = data categorized calculated or condensed to give meaning (sales report sales A > sales B)
- K = info synthesised & analyzed to find patterns rules connections
- the how (A > B on weekends cuz leisure item promoted in fri newsletters)
- W = Applying knowledge to make strategic judgements
- the why foresight ethics long term (move advertising budget → weekend)

importance

- global volume of data from enterprises etc keeps growing & problems get more complex → automation
- hand-written rules = costly & error prone
- humans provide data extract features
- the algorithm figures out its rules on its own to produce the result

non-ML vs ML

- both have the same pipeline dataset → feature extraction → rule → result
- the only difference is who writes the rule
 - non-ML → human
 - issue there r too many possible cases
 - eg IF smooth & orange THEN orange
 - IF smooth & red THEN apple
 - ML = give the algorithm lots of examples & it figures out the rules by itself
- algorithm = the code the programmer writes
- model = what comes out after the algorithm learns from data

categories

supervised

- every example has the correct label (photos tagged cat or dog)
- classification = predict a specific class label from predefined data
- which one is it or what kind is it
 - determining if a financial profile falls to bad or fair or good

- IF $x > \theta_1$ AND $y > \theta_2$ THEN A
- the threshold numbers are the discriminants
- IF temperature > 31 AND humidity $> 90\%$ THEN sunny $\rightarrow 31$ & 90% are the discriminants
- IF payment history $< 45\%$ AND length of credit history $< 30\%$ AND type of credit $< 50\%$ THEN bad
- regression = predict a continuous real number
- how much or how many
 - $y = wx + w_0$
 - predict ML score from Statistics Discrete Math & Programming scores $\rightarrow$ find the weight of each

classification	regression
category / label	a specific number
which bucket?	how much?
pass / fail	student's exact final score

unsupervised

- no labels
- machine needs to find hidden structures in messy data
- never looks at target y only studies how features relate to each other
- Clustering (grouping profiles) see overall characteristics of ppl or things organize to distinct segments based on how similar they are
 - which of these things belong together in a group eg young parents // healthy eaters
 - grouping customers by their past transactions
 - finding similar pictures and categorizing news
 - clustering cuz nobody gave the groups a name beforehand
- association (connecting items rules) finds specific triggers or links between items or events that freq happen at the same time
 - if X happens what else is likely to happen eg if buy A 70% chance buy B
 - = basket analysis
- dimensionality reduction $\rightarrow$ squeezes many features down to fewer
 - eg dataset of students each student 50 features (scores in 50 subjects)

most of the features move together (if good at calculus also good at alin etc)

 - $50 \rightarrow 3$ math ability language ability creative ability
 - fewer features = faster training less memory less noise $\rightarrow$ prevent overfitting easier visualisation (plot on 2D chart)
- clustering vs dimensionality reduction

- clustering $\rightarrow$ rows (groups the samples)
- group segment similar customers
- dimensionality reduction $\rightarrow$ columns (fewer features)
- too many features compress visualize in 2D or 3D

other types

type	answer key	example
supervised	all labelled	photos tagged cat or dog
unsupervised	no labels	grouping customers
semi-supervised	some labels	100 tagged + 10000 untagged
reinforcement	rewards	game AI +1 win and -1 lose

- semi-supervised (leans more to supervised) labelling data is expensive so only label a small part $\rightarrow$ model learns from these labels (like Google Photos)
- reinforcement = agent takes actions in an environment & gets reward or penalty $\rightarrow$ over time learns which actions earn most reward

$P(Y|X)$

- the probability of Y given X
- $P(\text{spread} | \text{bread}) = 0.7 \rightarrow 70\%$ of customers who buy bread also buy choc spread
- $P(Y | X, D) =$ of the customers who are young & single (D) and bought bread (X), the fraction who also bought choc spread (Y)
- probability can never be > 1
- $P(Y | X) = P(X \text{ and } Y) / P(X)$
 - 100 customers buy bread and 70 of them also buy spread $\rightarrow 70 / 100 = 0.7$

terms

- column = feature = independent variable (X)
- row = sample = one data point
- target = y = dependent variable
- Boston table one row = a sample (one house area in Boston)
- target = house price $\rightarrow$ what were predicting
- python libs = NumPy (arrays) SciPy (numerical algorithms) matplotlib (plots) pandas (tables) scikit-learn (ML algorithms)
- no notebook this week

exam check

- E T P for a scenario
 - spam filter $\rightarrow E$ = mails marked spam or not and T = decide if a new mail is spam and P = share it gets right

- which task and why
 - price of a used car $\rightarrow$ regression cuz the answer is a number
 - pass or fail $\rightarrow$ classification cuz the answer is a label
 - customer groups with no names $\rightarrow$ clustering cuz there are no labels
 - people who buy bread also buy spread $\rightarrow$ association cuz it links things that happen together
- supervised or not $\rightarrow$ does the data have the right answer
- AI or ML $\rightarrow$ is anything learned from data
- mix ups
 - classification and regression are both supervised
 - clustering groups rows and dimensionality reduction shrinks columns
 - algorithm = the code and model = what the code learned
 - feature = column = X and target = Y

week 2

Regression

- supervised algorithm that predicts a continuous value from one or more variables
 - house price 24.0 $\rightarrow$ regression cuz the answer is a number on a scale
 - spam or not spam $\rightarrow$ not regression cuz the answer is a label
- X = independent variables = features
- Y = dependent variable = target
 - Boston table $\rightarrow$ the 13 columns CRIM to LSTAT are X and house price is Y

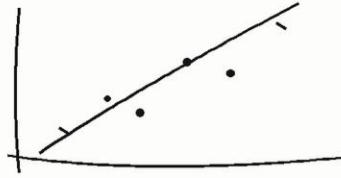
task	features	target
weather	temperature humidity pressure	wind speed
housing	rooms transport land size	house price
stocks	profit sector month economy	stock price
gasoline	weight horse power engine	MPG

- all four are regression cuz every target is a number

Simple linear

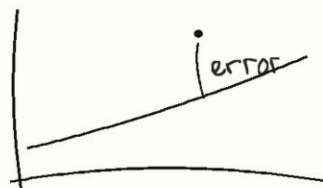
- $\hat{y} = \theta_0 + \theta_1 x_1$
- $\hat{y}$ = predicted variable
- θ_0 = intercept $\rightarrow$ the value of $\hat{y}$ when x is 0
- θ_1 = slope $\rightarrow$ how much $\hat{y}$ changes when x goes up by 1
- x_1 = the single predictor
 - $\hat{y} = 1.0 + 1.7x \rightarrow$ at $x = 2$ we get $\hat{y} = 4.4$

- slide example $\rightarrow x_1 = 5.631$ rooms
- training = find the θ whose line fits the points best



error and metrics

- error = $y - \hat{y}$
 - real $y = 16.5$ and predicted $\hat{y} = 15 \rightarrow$ error = 1.5
- $MSE = (1/n) \sum (y - \hat{y})^2$
- squaring $\rightarrow$ plus and minus errors cannot cancel and big errors are punished more
- objective of linear regression = find the θ that minimise the MSE
- MAE = mean of $|y - \hat{y}|$
 - $MAE = (1/n) \sum |y - \hat{y}|$
 - errors $0.3 - 0.4 - 0.1 0.2 \rightarrow MAE = 1.0 / 4 = 0.25$
 - same units as y so it reads as off by 0.25 on average
- MSE = mean of the squared errors
 - same errors $\rightarrow 0.30 / 4 = 0.075$
 - one big miss dominates it cuz of the square
- RMSE = $\sqrt{MSE} \rightarrow$ back in the units of y
 - $\sqrt{0.075} = 0.27$
- SSR = sum of squared residuals
 - $SSR = \sum (y - \hat{y})^2$
 - $MSE = SSR / n$
 - $0.30 / 4 = 0.075$
- R^2 = share of the variation in y that the model explains
 - $R^2 = 1 - \sum (y - \hat{y})^2 / \sum (y - \bar{y})^2$
 - least squares example $\rightarrow 1 - 0.30 / 14.75 = 0.98$
 - 1 = perfect
 - 0 = no better than always guessing the mean
 - 0.73 = explains 73% of the variation



least squares

- $x = 1 2 3 4$ and $y = 3 4 6 8 \rightarrow \bar{x} = 2.5$ and $\bar{y} = 5.25$
- $\theta_1 = \sum (x - \bar{x})(y - \bar{y}) / \sum (x - \bar{x})^2$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

x	y	x - $\bar{x}$	y - $\bar{y}$	product	(x - $\bar{x}$) ²
1	3	-1.5	-2.25	3.375	2.25
2	4	-0.5	-1.25	0.625	0.25
3	6	0.5	0.75	0.375	0.25
4	8	1.5	2.75	4.125	2.25
sum				8.5	5

$$\theta_1 = 8.5 / 5 = 1.7$$

$$\theta_0 = 5.25 - 1.7 \times 2.5 = 1.0$$

$$\hat{y} = 1.0 + 1.7x$$

x	y	$\hat{y}$	error	error ²
1	3	2.7	0.3	0.09
2	4	4.4	-0.4	0.16
3	6	6.1	-0.1	0.01
4	8	7.8	0.2	0.04

$$MSE = 0.30 / 4 = 0.075$$

multiple linear

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$$\hat{y} = \theta^T X \quad (x_0 = 1)$$

- θ for many features in one step = the normal equation

$$\theta = \text{inverse of } (X^T X) \times X^T y$$

$$\text{Price} = \theta_0 + \theta_1 \text{CRIM} + \theta_2 \text{LSTAT} + \theta_3 \text{RM} + \theta_4 \text{DIS}$$

- same idea with one θ per feature

- slide example $\rightarrow \theta = [-4.56 \quad -0.09 \quad -0.55 \quad 5.47]$

- row 8 $\rightarrow$ CRIM 0.21124 and LSTAT 29.93 and RM 5.631

$$\text{price} = -4.56 - 0.09 \times 0.21124 - 0.55 \times 29.93 + 5.47 \times 5.631$$

$$\text{price} = -4.56 - 0.019 - 16.46 + 30.80 = 9.76$$

- real price 16.5 $\rightarrow$ error = 6.74

- reading the signs

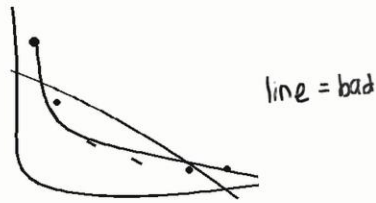
- minus on CRIM and LSTAT $\rightarrow$ more crime or more low status people means lower price
- plus on RM $\rightarrow$ more rooms means higher price

nonlinear

- use it when the data cannot be fit by a straight line

- price vs crime rate drops fast then flattens $\rightarrow$ a line misses the bend

- shapes = linear and quadratic and cubic and polynomial



polynomial

$$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

- trick $\rightarrow$ rename $x_1 = x$ and $x_2 = x^2$ and $x_3 = x^3$

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

- now it is multiple linear regression so least squares solves it

- goal = minimise the sum of squared residuals between y and $\hat{y}$

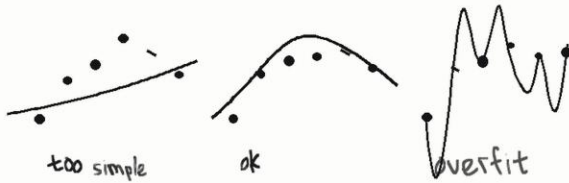
- overfitting = the model memorises the training points and fails on new ones

- degree 10 curve through 12 points $\rightarrow$ train error near 0 and test error large

- sign $\rightarrow$ train error falls while test error rises

- underfitting = the model is too simple for the pattern

- straight line on a curved cloud $\rightarrow$ both train and test error stay high



actual non-linear

- $\hat{y}$ is a non-linear function of the parameters θ

- how to tell $\rightarrow$ can I rename the x terms and get a plain sum of θ times something

- $\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 \rightarrow$ yes so it is still linear in θ

- $\hat{y} = \theta_0 + \theta_2^2 x \rightarrow$ no cuz θ_2 is squared

- $\hat{y} = \theta_0 + \theta_1 \theta_2^2 x \rightarrow$ no cuz two θ are multiplied

- $\hat{y} = \log(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \rightarrow$ no cuz θ sits inside the log

- $\hat{y} = \theta_0 / (1 + \theta_1^{\wedge}(x - \theta_2)) \rightarrow$ no cuz θ is in a power and a division $\rightarrow$ S curve

- no formula gives θ directly $\rightarrow$ found by repeating small improvements

regularization

- prevents overfitting by adding a penalty on large coefficients

$$\text{Ridge} = \text{MSE} + \alpha(\theta_1^2 + \theta_2^2 + \dots)$$

$$\text{Lasso} = \text{MSE} + \alpha(|\theta_1| + |\theta_2| + \dots)$$

- Ridge = squared penalty $\rightarrow$ shrinks every coefficient toward 0 but never to 0

- use when many features are highly correlated
- 10 near duplicate columns $\rightarrow$ Ridge keeps all 10 with small coefficients
- Lasso = absolute penalty $\rightarrow$ forces some coefficients to exactly 0 $\rightarrow$ feature selection
 - use when only a few features matter
 - same 10 columns $\rightarrow$ Lasso keeps 2 or 3 and zeroes the rest
- α = penalty strength
 - $\alpha = 0 \rightarrow$ plain linear regression
 - α very big $\rightarrow$ all coefficients near 0 $\rightarrow$ underfit
 - pick α with GridSearchCV
- CAREFUL the slide writes Ridge (L1) and Lasso (L2)
- the standard naming is Ridge = L2 and Lasso = L1
- if asked $\rightarrow$ describe what each one does as well as the name

good practice

- not on the slides
- look first $\rightarrow$ scatter each feature vs the target and spot curves and outliers
- split with a seed so results repeat
`train_test_split(X, y, test_size=0.2, random_state=42)`
- baseline $\rightarrow$ always predicting the mean gives $R^2 = 0$ and the model must beat it
- start with a line then check the residual plot of error vs y
 - random cloud $\rightarrow$ fine
 - curve $\rightarrow$ wrong shape so try polynomial
 - funnel $\rightarrow$ the error grows with y
- pick the degree with cross validation on train and never by the test score

$$cv\ score = (score_1 + score_2 + \dots + score_n) / n$$
 - 5 folds $\rightarrow$ train on 4 and score on 1 and do it 5 times
- Ridge and Lasso $\rightarrow$ scale the features first and fit the scaler on train only

$$z = (x - \text{mean}) / \text{std}$$

$$x = 30 \text{ with mean } 20 \text{ and std } 5 \rightarrow z = 2$$
- alpha $\rightarrow$ search powers of 10 then use the alpha the search returns
- leakage = anything fitted on test rows $\rightarrow$ a Pipeline prevents it
`make_pipeline(StandardScaler(), PolynomialFeatures(2), Ridge(alpha=1))`
- coefficient sizes are only comparable when features are scaled
- which model
 - line first $\rightarrow$ polynomial if the residuals curve $\rightarrow$ Ridge if many correlated features $\rightarrow$ Lasso to drop useless ones

exam check

- predict by hand $\rightarrow$ plug the row into $\hat{y} = \theta_0 + \theta_1 x_1 + \dots$ and add the terms
- error = real minus predicted and MSE = mean of the squared errors
- find θ by hand $\rightarrow$ the two least squares formulas then the table
- linear or not $\rightarrow$ check if it is a plain sum of θ times something
- overfit or underfit $\rightarrow$ compare train error with test error
- Ridge or Lasso $\rightarrow$ shrink all vs zero some
- regression or classification $\rightarrow$ number vs label
- mix ups
 - MAE vs MSE $\rightarrow$ absolute vs squared
 - R^2 is a share and not an error $\rightarrow$ higher is better
 - $\text{coef}_1 = \text{slopes}$ and $\text{intercept}_1 = \theta_0$
 - fit_transform on train and transform only on test
 - fit learns θ and predict uses θ

lab code

- Boston house prices with scikit-learn
- every model follows the same order $\rightarrow$ make the model then fit then predict then score

load

```
raw_df = pd.read_csv(data_url, sep="\t", skiprows=22,
                     header=None)
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]
boston_df = pd.DataFrame(np.c_[data, target], columns=names)
```

1 `pd.read_csv` $\rightarrow$ reads the file into a table $\rightarrow$ `sep` splits on spaces and `skiprows` skips the text at the top

2 `np.hstack` $\rightarrow$ joins two arrays side by side $\rightarrow$ each town sits on 2 lines so this makes one row of 13 features

3 `values[1::2, 2]` $\rightarrow$ every second line and its third number $\rightarrow$ the house price

4 `np.c_` $\rightarrow$ sticks the target on as one more column

5 `pd.DataFrame` $\rightarrow$ turns the array into a table with column names

- CAREFUL `sep="\t"` gives a SyntaxWarning and the correct form is `sep=r"\t"`

explore

```
boston_df.describe()
plt.scatter(bdf.CRIM, bdf.target)
```

```
bdf[['CRIM', 'ZN']]hist()
```

- 1 describe $\rightarrow$ Count mean std min and max of every column
- 2 plt.scatter $\rightarrow$ dots of crime rate against price $\rightarrow$ shows the price drops fast then flattens
- 3 hist $\rightarrow$ bar chart of how the values of a column are spread

split

```
dat = np.random.rand(len(boston_df)) < 0.8
```

```
train = bdf[dat]
```

```
test = bdf[~dat]
```

- 1 np.random.rand $\rightarrow$ one random number from 0 to 1 per row
 - 2 $< 0.8 \rightarrow$ True for about 80% of the rows
 - 3 bdf[dat] $\rightarrow$ keeps the True rows $\rightarrow$ training set
 - 4 bdf[~dat] $\rightarrow$ ~ means NOT so it keeps the other rows $\rightarrow$ test set
- CAREFUL no seed so the split and every score change on each run

simple linear

```
regr = linear_model.LinearRegression()
```

```
train_x = np.asanyarray(train[['CRIM']])
```

```
train_y = np.asanyarray(train[['target']])
```

```
regr.fit(train_x, train_y)
```

```
print(regr.coef_, regr.intercept_)
```

```
test_y_ = regr.predict(test_x)
```

```
np.mean(np.absolute(test_y_ - test_y))
```

```
np.mean((test_y_ - test_y) ** 2)
```

- 1 LinearRegression $\rightarrow$ makes an empty model
 - 2 np.asanyarray $\rightarrow$ turns the column into an array $\rightarrow$ double brackets
- Keep it 2-D which fit needs
- 3 fit $\rightarrow$ least squares finds the best θ from the training rows
 - 4 coef_ $\rightarrow$ the slope θ_1 and intercept_ $\rightarrow \theta_0$
 - 5 predict $\rightarrow$ $\hat{y}$ for rows the model has not seen
 - 6 np.mean of np.absolute $\rightarrow$ MAE
 - 7 np.mean of the squared difference $\rightarrow$ MSE
- lab $\rightarrow$ coef -0.408 and intercept 23.99 and MAE 5.92 and MSE 63.20

multiple linear

```
x = np.asanyarray(train[['CRIM', 'LSTAT', 'RM']])
```

```
regr.fit(x, y)
```

```
regr.score(test_x, test_y)
```

- 1 three column names $\rightarrow$ X now has 3 features so the model learns 3 slopes
 - 2 fit $\rightarrow$ same call and nothing else changes
 - 3 score $\rightarrow$ R^2 on the test rows
- lab $\rightarrow$ coeffs -0.094 and -0.606 and 5.07 with intercept -1.52 and R^2 0.73

- CAREFUL the lab prints the mean squared error under the label Residual sum of squares

polynomial

```
poly = PolynomialFeatures(degree=2)
train_x_poly = poly.fit_transform(train_x)
clf = linear_model.LinearRegression()
clf.fit(train_x_poly, train_y)
test_y_ = clf.predict(poly.transform(test_x))
```

- 1 PolynomialFeatures → a tool that builds the power columns
- 2 fit_transform → turns the column x into 1 and x and x^2
- 3 LinearRegression and fit → an ordinary linear model on those columns
→ a curve in x
- 4 transform → builds the same columns for the test rows without fitting again
- 5 predict → $\hat{y}$ from the curve
 - the first coefficient is always 0 cuz the column of ones repeats the intercept
 - lab degree 3 → MAE 5.33 and MSE 54.19 which beats the line at 63.20

Ridge and Lasso

```
parameters = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
ridge_regressor = GridSearchCV(Ridge(), parameters,
                                scoring='neg_mean_absolute_error', cv=5)
ridge_regressor.fit(train_x, train_y)
print(ridge_regressor.best_params_, ridge_regressor.best_score_)
ridge_model = Ridge(alpha=100)
ridge_model.fit(train_x, train_y)
test_y_ = ridge_model.predict(test_x)
```

- 1 parameters → the penalty strengths to try
 - 2 GridSearchCV → tries every alpha with 5 fold cross validation →
cv=5 splits train into 5 parts and trains on 4 and scores on 1
 - 3 scoring → negative MAE cuz sklearn always picks the highest score
 - 4 best_params_ → the alpha that scored best and best_score_ → its score
 - 5 Ridge(alpha=100) → a new model with a chosen alpha
 - 6 fit and predict → as before
 - Lasso → the same lines with Lasso() in place of Ridge()
 - lab → Ridge search picks 0.001 then MAE 7.42 and Lasso search picks $1e-16$ then MAE 7.39
- CAREFUL the lab trains with alpha 100 and alpha 1 and not the alpha the search picked
 - CAREFUL those scores come from a different random split so do not compare them with 5.92

Week 3

time series

- data in time order → the model reads the past and predicts future values
- every row has a timestamp
- it tracks variables across time
- points depend on each other so never shuffle the rows
 - monthly sales Jan to Dec → time series cuz the order carries the pattern
 - heights of 30 students → not a time series cuz the order means nothing
- univariate = time is the only independent variable
 - monthly sales alone → univariate cuz we predict sales from its own past
- multivariate = more than one independent variable
 - sales plus weather and holidays → multivariate cuz other columns help predict
- use cases
 - forecasting = predict future values from history
 - next months sales or traffic in 30 minutes → forecasting cuz the answer is a future value
 - anomaly detection = detect unusual behaviour
 - odd network activity or a strange transaction → anomaly cuz it breaks the usual pattern
 - pattern analysis = find trends and seasonal effects and changes
 - higher sales on weekends or peak hours of power use → pattern cuz we describe what repeats

autocorrelation

- how a series relates to its own past values
- tells if past observations hold useful information for predicting the future
- lag k = k steps back → lag 2 = the value 2 days before
- correlation between two columns x and y
$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{(\sum (x - \bar{x})^2 \times \sum (y - \bar{y})^2)}}$$
 - from -1 to 1 and 0 means no straight line link
- ACF at lag k = correlation between $y(t)$ and $y(t-k)$
 - a tall bar at lag 12 on monthly data → a yearly season cuz this month looks like the same month last year
- PACF = link between $y(t)$ and $y(t-k)$ after removing what the lags in between already explain

- Jan affects Mar only through Feb $\rightarrow$ ACF at lag 2 is high and PACF at lag 2 is near 0
- ACF = total link and PACF = direct link only
- $PACF(1) = ACF(1)$
- $PACF(2) = (r_2 - r_1^2) / (1 - r_1^2)$
- $r_1 = 0.385$ and $r_2 = 0.269 \rightarrow (0.269 - 0.148) / (1 - 0.148) = 0.142$
- $ACF(k) = \sum dev(t) \times dev(t-k) / \sum dev^2$
- dev = value minus the mean
- which bars count $\rightarrow$ only the ones outside the shaded band
- band = $1.96 / \sqrt{n}$
- 48 months $\rightarrow$ band = 0.28
- lag 0 is always 1 so ignore it
- by hand $\rightarrow 12\ 15\ 14\ 18\ 20\ 19\ 23$ with mean 17.29
- deviations $\rightarrow -5.29\ -2.29\ -3.29\ 0.71\ 2.71\ 1.71\ 5.71$
- top = $\sum dev(t) \times dev(t-1) = 33.63$
- bottom = $\sum dev^2 = 87.43$
- $ACF(1) = 33.63 / 87.43 = 0.385$

trend seasonality cycle

- trend = long term increase or decrease
 - an online store slowly sells more as its Customer base grows $\rightarrow$ trend cuz it keeps going one way
- seasonality = a pattern that repeats at a predictable fixed interval
 - hourly daily weekly monthly quarterly or yearly
 - electricity use up every evening $\rightarrow$ seasonal cuz the gap is always one day
 - supermarket sales up every weekend and hotel bookings up every holiday and ice cream up every summer
 - airline passengers peak every year
- cycle = rises and falls whose timing and length are not fixed
 - economic expansions and recessions $\rightarrow$ cycle cuz nobody can say how long each one lasts
- test $\rightarrow$ can I say how many steps until it repeats
 - yes $\rightarrow$ seasonality
 - no $\rightarrow$ cycle



stationarity

- the statistical characteristics stay stable over time
- three conditions

- constant mean
- constant variance
- covariance depends on the lag and not on the time

$$\text{mean} = (1/n) \sum y$$

$$\text{variance} = (1/n) \sum (y - \bar{y})^2$$

$$\text{covariance at lag } k = (1/n) \sum (y(t) - \bar{y})(y(t-k) - \bar{y})$$

- consumption that climbs every month $\rightarrow$ not stationary cuz the mean keeps moving

consumption that wobbles around one average with steady spread $\rightarrow$ stationary

- why it matters $\rightarrow$ ARIMA assumes a stationary series
- fix a trend by differencing $\rightarrow$ replace $y(t)$ with $y(t) - y(t-1)$

$$- 12 \ 15 \ 14 \ 18 \ 20 \ 19 \ 23 \rightarrow +3 \ -1 \ +4 \ +2 \ -1 \ +4$$

- the climb is gone and the values hover around 18

$$w(t) = y(t) - y(t-1)$$

$$\text{seasonal } w(t) = y(t) - y(t-s)$$

- ADF test checks it

- $p < 0.05 \rightarrow$ stationary
- lab $\rightarrow p = 0.0002$ so stationary
- $p = 0.40 \rightarrow$ not stationary so difference once and test again

$$y(t) - y(t-1) = c + g \times y(t-1) + e(t)$$

- the test asks if $g = 0$

- $g = 0 \rightarrow$ not stationary and g below 0 $\rightarrow$ stationary



ARIMA

- Auto Regressive Integrated Moving Average $\rightarrow$ uses past values to forecast the future

- p = autoregressive order = number of lags

- $p=2 \rightarrow$ today depends on the last 2 values

$$AR(p)y(t) = c + a_1 y(t-1) + a_2 y(t-2) + \dots + e(t)$$

- d = how many times the series is differenced to become stationary

- usually 1 to remove a trend

- q = moving average order = number of past errors used

- $q=1 \rightarrow$ today also corrects for the last forecast error

$$MA(q)y(t) = c + e(t) + b_1 e(t-1) + b_2 e(t-2) + \dots$$

- $e(t)$ = the error at time t = real minus forecast

- ARIMA(1, 1, 1) $\rightarrow$ 1 past value and differenced once and 1 past error

$$w(t) = y(t) - y(t-1)$$

$$w(t) = c + a_1 w(t-1) + e(t) + b_1 e(t-1)$$

- a = AR weights and b = MA weights and c = constant

- AIC = score for comparing models $\rightarrow$ fit quality with a penalty for extra parameters

- lower is better

- AIC 790 vs 1066 $\rightarrow$ pick 790

$$AIC = 2k - 2 \ln(L)$$

- k = number of parameters and L = likelihood = how well the model fits

- more parameters raise it and better fit lowers it

- steps for building it

- visualization

- ACF and PACF plot

- seasonal variation modelling

- stationary check

- identify p and q for the non seasonal part

SARIMA

- Seasonal ARIMA = ARIMA extended for data with seasonal patterns

handles periodic fluctuations so it forecasts seasonal data better

$$SARIMA(p, d, q)(P, D, Q, s)$$

- p and q = the non seasonal part as in ARIMA

- P = seasonal autoregressive order

- D = seasonal differencing order $\rightarrow y(t) - y(t-12)$

- Q = seasonal moving average order

- s = length of the season $\rightarrow 12$ for monthly data

- $SARIMA(1,1,1)(1,1,1,12)$ $\rightarrow$ the same three ideas again but looking 12 months back

$$w(t) = c + a_1 w(t-1) + b_1 e(t-1) + A_1 w(t-12) + B_1 e(t-12) + e(t)$$

- A and B = the seasonal weights

- components = seasonal and autoregressive and integrated and moving average

SARIMAX

- SARIMA with exogenous regressors $\rightarrow$ also uses external variables

- S = seasonal $\rightarrow$ periodic patterns like weekly or monthly or yearly cycles

- AR = autoregressive $\rightarrow$ link between the current value and previous values

- I = integrated $\rightarrow$ differencing to make the series stationary

- MA = moving average $\rightarrow$ the current value depends on past error terms

- X = exogenous regressors $\rightarrow$ external variables that may affect the series

- a holiday flag or a promo or the weather $\rightarrow X$ cuz it comes from outside the series

- it must be known for the months being forecast
- $x(t) = \text{SARIMA part} + \beta x(t)$
- $x(t)$ = the outside variable and β = its weight
- lab style flag $\rightarrow \beta$ = extra sales in a holiday month

which one

- no season $\rightarrow$ ARIMA
- season $\rightarrow$ SARIMA
- season plus a known outside variable $\rightarrow$ SARIMAX

seasonal index

- not on the slides
- how strong a period is against an average period $\rightarrow 100$ = average
- index = period average / overall mean $\times 100$
- sales $Y_1 = 2030$ 25 46 and $Y_2 = 24$ 36 28 52 and $Y_3 = 2840$ 31 59

quarter	average	index
Q1	24	69.06
Q2	36	100.72
Q3	28	80.58
Q4	52	149.64

- overall mean = 34.76
- Q4 index 149.64 $\rightarrow$ Q4 sells about 50% above an average quarter
- deseasonalised Q4 of $Y_3 = 59 / 1.4964 = 39.43$
- deseasonalised = actual / (index / 100)
- forecast = level $\times$ (index / 100)
- level 40 in Q4 $\rightarrow 40 \times 1.4964 = 59.9$

choosing p d q

- not on the slides
- d first $\rightarrow$ ADF on the series
 - $p < 0.05 \rightarrow d = 0$
 - else difference once and test again $\rightarrow$ rarely more than 1
- read ACF and PACF on the differenced series and not the raw one
- PACF cuts off after lag p and ACF fades slowly $\rightarrow$ AR(p) so p from PACF and $q = 0$
- ACF cuts off after lag q and PACF fades slowly $\rightarrow$ MA(q) so q from ACF and $p = 0$
- both fade slowly $\rightarrow$ mixed so start with (1, d, 1)
 - cuts off = the bars drop inside the band and stay there
 - fades = the bars shrink bit by bit
 - PACF bars outside the band at lag 1 and 2 only $\rightarrow p = 2$ and $q = 0$
- season $\rightarrow$ ACF spikes at 12 and 24 mean $s = 12$
 - seasonal spikes fade slowly $\rightarrow D = 1$
 - p from PACF at lag 12 and q from ACF at lag 12 $\rightarrow$ usually 0 or 1

- real plots are messy → use them for a first guess then try neighbours and compare AIC
 - only compare AIC between models with the same d and D
- check the fit → residuals should look like noise with no bar outside the band
- evaluate → split by time and hold out the last season and beat repeat last year
- the lab → ADF $p = 0.0002$ so $d = 0$ is enough and the grid pick of $d = 2$ over differences

exam check

- name the pattern → fixed gap is seasonality and no fixed gap is cycle and one direction is trend
- stationary or not → check mean and variance over time or read the ADF p value
- difference by hand → subtract the previous value
- ACF by hand → deviations then top over bottom
- read SARIMA(1,1,1)(1,1,1,2) → name each of the 7 numbers
- pick the model → season or not and outside variable or not
- seasonal index by hand → period average over overall mean times 100
- mix ups
 - ACF = total link and PACF = direct link
 - d removes a trend and D removes a season
 - q counts past errors and not past values
 - lower AIC is better and lower MAE is better
 - never shuffle and never split a time series randomly

lab code

- store sales with statsmodels

load

```
data_df = pd.read_csv(url)
```

```
sales_df = data_df[['Order Date', 'Sales']]
```

```
sales_df['Order Date'] = pd.to_datetime(sales_df['Order Date'])
```

1 `pd.read_csv` → reads the 9994 order rows

2 double brackets → keeps only the date and the sales columns

3 `pd.to_datetime` → turns the date text into real dates so they can be sorted and grouped

daily to monthly

```
daily_sales = sales_df.groupby('Order Date')['Sales'].sum().sort_index()
```

```
df1 = sales_df.set_index('Order Date')
```

```
monthly_sales = df1.resample('ME').sum()
```

- 1 groupby → puts rows with the same date together
- 2 sum → adds the sales of each date
- 3 sort_index → puts the dates in order
- 4 set_index → makes the date the row label which resample needs
- 5 resample('ME') → groups by month end and sum adds each month → 48 points

stationarity ACF PACF

```
result = adfuller(timeseries, autolag='AIC')
```

```
p_value = result[1]
```

```
plot_acf(monthly_sales)
```

```
plot_pacf(monthly_sales)
```

- 1 adfuller → runs the ADF test and returns several numbers
 - 2 result[0] → the test statistic and result[1] → the p value
 - 3 plot_acf → bar per lag showing the correlation with that lag
 - 4 plot_pacf → the same for the direct link only
- lab → statistic -4.49 and $p = 0.0002$ so stationary and the lag 12 bar stands out

ARIMA

```
pdq = list(itertools.product(range(0, 4), range(0, 3), range(0, 4)))
```

for order in pdq:

```
model = ARIMA(monthly_sales, order=order)
```

```
results = model.fit()
```

```
if results.aic < best_aic:
```

```
    best_aic = results.aic
```

```
    best_order = order
```

```
results = ARIMA(monthly_sales, order=best_order).fit()
```

```
forecast_values = results.forecast(steps=12)
```

- 1 itertools.product → every combination of p and d and q → $4 \times 3 \times 4 = 48$

- 2 ARIMA → makes the model for one order

- 3 fit → estimates it from the data

- 4 results.aic → its AIC and the loop keeps the lowest

- 5 forecast(steps=12) → predicts the next 12 months

- the lab wraps the fit in try and except so orders that crash are skipped

- lab → best order (3, 2, 3) with AIC 1065.76

- CAREFUL ADF said stationary yet the search picked $d = 2$

- CAREFUL the model is fitted on all 48 months so the forecast is for 2018

evaluate

observed = monthly_sales.tail(12)

mae = mean_absolute_error(observed, forecast_values)

mse = mean_squared_error(observed, forecast_values)

1 tail(12) → the last 12 rows → the months of 2017

2 mean_absolute_error → MAE between the real values and the forecast

3 mean_squared_error → MSE

- lab → MAE 43745

- CAREFUL observed is 2017 and the forecast is 2018 so this MAE is not a real accuracy

- correct way → fit on monthly_sales[:-12] then forecast 12 then compare with monthly_sales[-12:]

SARIMA

p, d, q = 1, 1, 1

P, D, Q, s = 1, 1, 1, 12

model = SARIMAX(monthly_sales, order=(p, d, q),
seasonal_order=(P, D, Q, s))

results = model.fit()

forecast = results.forecast(steps=12)

1 order → the normal part

2 seasonal_order → the seasonal part with s = 12 months

3 SARIMAX → the class used for SARIMA too

4 fit and forecast → as in ARIMA

- lab → MAE 10612 which is the best of the three

SARIMAX

model = SARIMAX(monthly_sales, order=(p, d, q), seasonal_order=(P,
D, Q, s), enforce_invertibility=stationarity,
enforce_stationarity=stationarity)

1 enforce_stationarity=False → lets the AR part be non stationary

2 enforce_invertibility=False → lets the MA part be non invertible

- lab → MAE 13031

- CAREFUL no exog is passed so this is SARIMA with two checks off and not a real SARIMAX

- a real one → SARIMAX(train, exog=holiday, ...) and forecast(steps=12, exog=future_holiday)

- CAREFUL the SARIMA and SARIMAX plots draw forecast_values which is the ARIMA forecast

- ranking → SARIMA 10612 then SARIMAX 13031 then ARIMA 43745