Python for Data Analytics

Module 5: Time Series & Forecasting

DeepLearning.AI

Time series & forecasting

Module 5 introduction

DeepLearning.AI

Module 5 outline

Stock prices

Weather

DateTime types

- Organize time series
- Line charts
- Multiple line charts

Descriptive statistics for time series

- Moving average
- Percent change
- Segmentation
- Reshaping & resampling

Time series forecasting

- Model trends
- Model seasonality
- Evaluate effectiveness

Time series & forecasting

DateTimes

DeepLearning.AI

Data types

In Python:

Numbers:

```
12, -1, 0.5, -15.5
```

Strings:

```
"Thank you."
```

In pandas:

Series:

```
s  = pd.Series([10, 20, 30, 40])
```

DataFrame:

```
df = pd.DataFrame({
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "City": ["NYC', 'LA", "Chicago"]  })
```

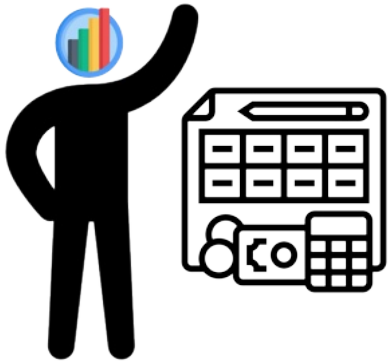
Both Python and pandas:

Dates:

```
dt = pd.to_datetime("2025-02-17 10:30:00")
```

```
# Output: 2025-02-17 10:30:00
```

Scenario



You
Data Analyst

🎯 **Task**: Looking for trends in tech and retail stocks before and after the COVID-19 pandemic

🏆 **Goal**: Advise clients on which stocks were most resistant to such a catastrophic event

📊 **Dataset**:

- Date
  - In YYYY-MM-DD format
  - Allows sorted in historical order
  - Not all dates are represented
- Ticker (nickname)
- Open, high, low, close, and adjusted close price
  - Price of stock at the end of day, adjusted for specific financial circumstances
- Trading volume
- Sector

Recap: DateTime

- To convert a string or a Series of strings into DateTime format:

```
date = pd.to_datetime("2035-02-17")
```

Method

```
date.weekday()
```

Attributes

```
date.month
```

```
date.day
```

```
date.year
```

```
date.quarter
```

- To access attributes of entire series, use the `.dt` accessor in between:

```
df["date"].dt.day
```

Time series & forecasting

Using DateTimes as indices

DeepLearning.AI

Recap: DataTimes as indices

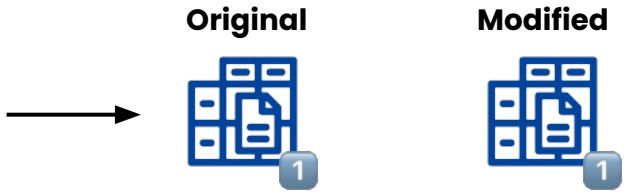
Default:

Column name

```
apple_stocks.set_index("datetime")
```

Override:

```
apple_stocks.set_index("datetime", inplace = True)
```



Select single row with a single date

```
apple_stocks.loc['2024-03-01']
```

Slice dataframe to select between two dates

```
apple_stocks.loc['2024-03-01':'2024-03-31']
```

- Includes both first and last indices

Time series & forecasting

Line charts

DeepLearning.AI

Recap: Line charts

- To create a line chart of time series data

```
sns.lineplot(apple_stocks_feb_2020["adjusted_close"])
```

- Use x and y named arguments for more control over data

```
sns.lineplot( x = apple_stocks_feb_2020.index.day,
              y = apple_stocks_feb_2020["adjusted_close"],
              color = "darkgray")
```

- To add a label to each line

```
sns.lineplot( x = apple_stocks_feb_2020.index.day,
              y = apple_stocks_feb_2020["adjusted_close"],
              color = "darkgray",
              label = "February 2020")
```

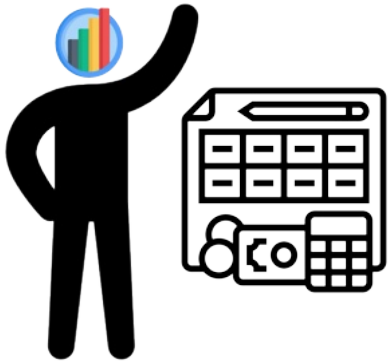
- Use the `index` to plot different values on the x axis like day of the month

Time series & forecasting

Formatting date axis labels

DeepLearning.AI

Scenario



You
Data Analyst

🏆 **Goal**: Investigating stocks to identify those that are resistant to large market shocks like the pandemic

🎯 **Task**: Look at Amazon stock price over time to better understand its performance around the pandemic

⬜ Visualize the stock price

Recap: Formatting date axis labels

- To work with dates, import:

```
import matplotlib.dates
```

- To mark each year on the x axis:

```
ax.xaxis.set_major_locator()
```

- To control how dates were formatted:

```
ax.xaxis.set_major_formatter()
```

Date format codes:

- `%y` → Year without century ('24)
- `%Y` → Full year (2024)
- `%m` → Zero-padded month (02)
- `%d` → Day (26)
- `%b` → Abbreviated month (Feb)

Combine together:

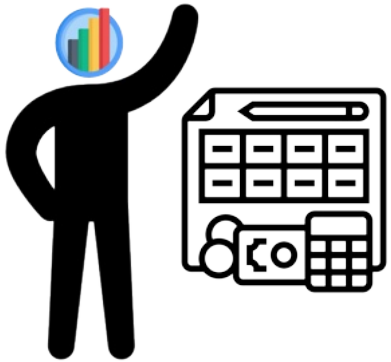
- `%m %y` → `02 24` (Month & Year)

Time series & forecasting

Moving average

DeepLearning.AI

Scenario



You
Data Analyst

🏆 **Goal**: To assess a stock's resistance to market shocks

- Look at its trading volume
  - Number of shares in the stock that were traded on that particular day.

🎯 **Task**: Analyze the trading volume of Amazon stocks around the time of the COVID-19 pandemic

Recap: Moving average

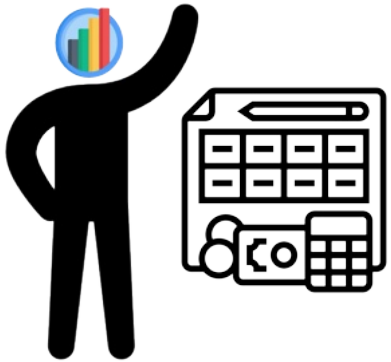
1. Select the column: `df["Volume"]`
2. Apply `.rolling()` with a window size: `df["Volume"].rolling(window=14)`
3. Use an aggregate function:
   - Moving average: `df["Volume"].rolling(window=14).mean()`
   - Different aggregation: `df["Volume"].rolling(window=14).median()`
4. Plot newly generated series: `sns.lineplot(amazon_volume_14day, color="black")`

Time series & forecasting

Percent change

DeepLearning.AI

Scenario



You
Data Analyst

🔍 **Discovery**: Amazon's trading volume spiked during the year 2020

🎯 **Task**: To identify dates with a negative price change of 5% or more

- Dips can help you identify market events that created largest price shocks over the years

Recap: Percent change

- To create a series representing percent change from period to period:

```
amazon_close_pct_change = amazon_stocks["adjusted_close"].pct_change()
```

- To graph percent change over time:

```
sns.lineplot(amazon_close_pct_change, color = "#FF9900")
```

- Filter data based on:
    - Positive or negative percent change
    - Certain magnitude of percent change

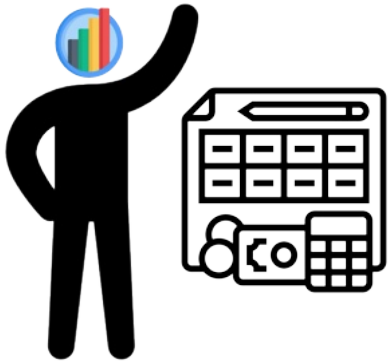
```
dips = amazon_close_pct_change[amazon_close_pct_change < -0.05]
```

Time series & forecasting

Segmentation

DeepLearning.AI

Scenario



You
Data Analyst

🏆 **Goal**: To investigate the resilience of different stocks to market shocks

🎯 **Task**: Investigate seasonalities to understand how companies are performing monthly and quarterly

📊 **Data**: Include streaming service Netflix

Recap: Segmentation

- To access aspects of datetime in time series:

```
netflix_stocks.index.year
```

Data frame — **Attribute to access**

- Save parts of the date to a new column:

```
netflix_stocks["quarter"] = netflix_stocks.index.quarter
```

- Group data by that feature of the date:

```
monthly_volume = netflix_stocks.groupby("month")["volume"].mean()
```

Time series & forecasting

Multiple line charts: reshaping

DeepLearning.AI

Multiple line charts - reshaping

- Used two calls to `sns.lineplot()` to plot multiple series together
  - Worked because with only a couple of stocks
  - Can get tedious and error prone

⛔ Creates duplicate work

⛔ Having to manually filter your data multiple times

- To create a more complex plot:
  - Reshape your data
  - Each value on the y axis is associated with the date time

✅ More efficient

Multiple line charts - reshaping

	date	ticker	open	high	low	close	adjusted_close	volume	sector
461	2015-10-30	AAPL	30.2475	30.3050	29.8625	29.8750	26.979687	197461200	Electronics
19474	2015-07-27	AMZN	26.3875	27.2475	26.3300	26.5705	26.570500	149820000	Retail
20699	2020-06-08	AMZN	125.0100	126.5000	124.3670	126.2030	126.203000	79414000	Retail
21323	2022-11-28	AMZN	93.9300	96.4000	93.4300	93.9500	93.950000	74943100	Retail
21140	2022-03-08	AMZN	136.6835	140.6995	133.5725	136.0145	136.014500	91662000	Retail
21743	2024-08-01	AMZN	189.2900	190.6000	181.8700	184.0700	184.070000	70435600	Retail
232	2014-12-03	AAPL	28.9375	29.0875	28.7775	28.9825	25.845362	172253600	Electronics
694	2016-10-04	AAPL	28.2650	28.5775	28.1575	28.2500	26.057646	118947200	Electronics
19788	2016-10-21	AMZN	40.4680	40.9710	40.4500	40.9495	40.949500	55860000	Retail
2630	2024-06-14	AAPL	213.8500	215.1700	211.3000	212.4900	212.244340	70122700	Electronics

✅ Reshape the data

✅ One row for each unique datetime

Recap: Multiple line charts - reshaping

- Use .isin() method to filter for multiple values

```
tickers = ["AAPL", "AMZN"]
selected_stocks = df[df["ticker"].isin(tickers)]
```

- Pivot (multiple y values for each x value)

```
pivoted_stocks = selected_stocks.pivot( index = "datetime",
                                        columns = "ticker",
                                        value = "adjusted_close" )
```

- Plot

```
sns.lineplot( pivoted_stocks )
```

Time series & forecasting

Resampling

DeepLearning.AI

Resampling

- **Previous**: Extract information about dates and save into new column
  - ✅ Useful for segmentation
  - ⛔ Doesn't make data any more or less frequent

- To get more or less aggregated periods, **resample** data:
  - 🔽 **Downsampling** - aggregating periods together
    - Example: Daily to weekly
  - 🔼 **Upsampling** - increasing frequency by creating new periods

Recap: Resampling

- To downsample your data:
  - `"W"` - Weekly
  - `"D"` - Daily
  - `"ME"` - Month End
  - `"W-MON"` - Weekly on Monday

```
nike_stocks.resample("W")
```

- Resampling mean only works on numeric columns:

```
nike_weekly = numeric_data.resample("ME").mean()
```

- Use different aggregation function for categorical data:

```
nike_weekly = numeric_data.resample("W").first()
```

Time series & forecasting

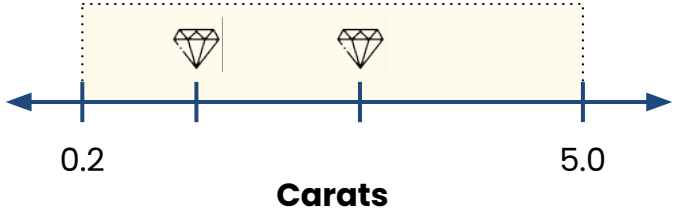
Forecasting with the trend

DeepLearning.AI

Time series

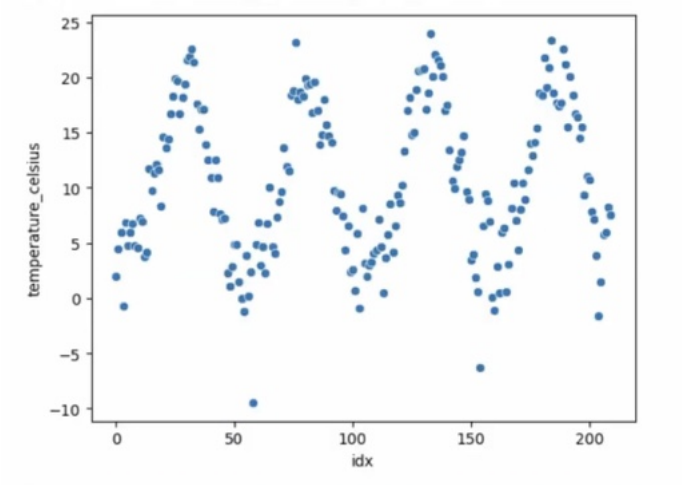
With the diamond data:

- Had bounds on possible values for independent and dependent variables



Linear regression to predict time series:

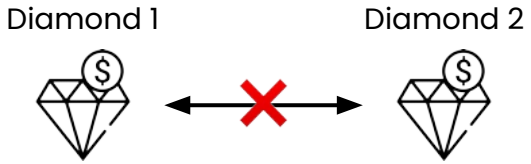
- Focus analysis on forecasting future periods
- Assumes trend and seasonality continue in future
- Increasing uncertainty the further you forecast
- Conditions change and small uncertainties compound over many years



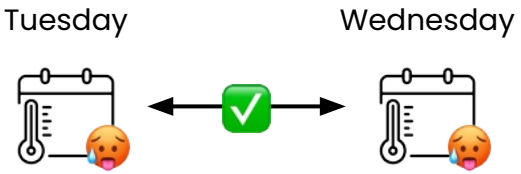
Complications

- Linear regression assumes independent observations
- Model will **overestimate** confidence for p values and coefficients
- For this reason:
  - Linear regression is a preliminary method
  - Some advanced methods compensate for inherent lack of independence

Price of diamond



Temperature



Time series forecasting

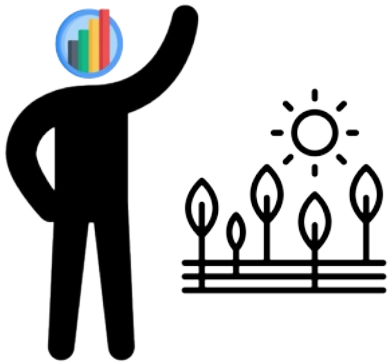
1️⃣ To reduce the issue with independence, resample your data

- ⏰ **Example**: From hourly to weekly
- ⬇️ Reduces overlapping noise

2️⃣ Assign an index number to each period to model the trend

- **Reason**: Linear regression model only works with numbers, not dates

Scenario



You
Data Analyst

🏆 **Goal**: Predict temperatures in Beutenberg, Germany to:

- Plan growing seasons for local farmers
- Better predict harvest profits

🎯 **Task**: Develop a model to predict future temperatures based on past data

- Focus on the time series data
- Won't see a train/test split, but it's good practice

Time series & forecasting

Forecasting with seasonality

DeepLearning.AI

Recap: Seasonality

- You can add seasonality to time series model using multiple linear regression
- To forecast new week using a higher `idx` than the values model trained on

```
week210 = [1, 210, 0, 0, 1]
results.predict(week210)

week210 = [1, 210, 0, 1, 0]
results.predict(week210)
```

- Seasonality is more than just seasons
  - Any **regular repeating pattern** in data
  - Weather is just one example
    - Weather patterns are so strongly predicted by the season

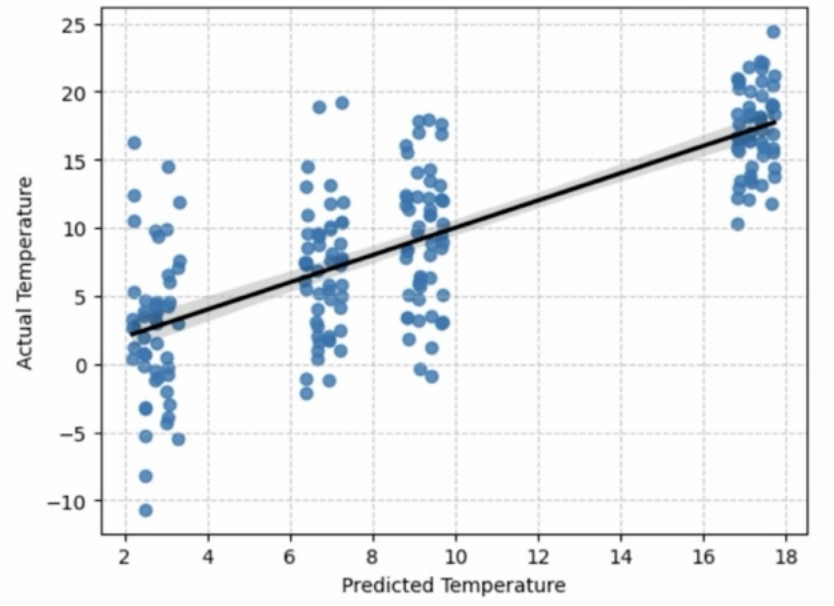
Time series & forecasting

Error metrics for forecasting

DeepLearning.AI

Recap: Error metrics

- Look at scatter plot to:
  - ⬜ Identify any nonlinear relationships
  - ⬜ See how model is performing
- Calculate the residuals
  - ⬜ Use to calculate mean absolute error
  - ⬜ Tells you on average how off model's predictions were
- Now that you've developed a model:
  - Bring model back to your client
  - Improve it using the iterative process




```
residuals = y_actual - y_pred

MAE = residuals.mean().abs()
```

Python for Data Analytics

Your next steps

DeepLearning.AI