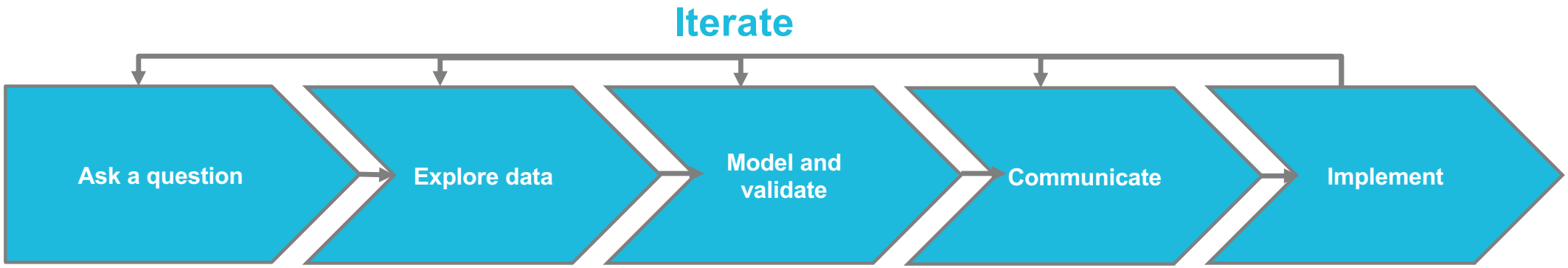
Feature Engineering

- Data Science process
- Feature engineering - overview
- Feature encoding
- Feature Selection

“Applied Machine Learning is basically Feature Engineering”

Andrew Ng

Data Science Process



- Ask a (business) question
- Define objective, strategy, value and effort budget.

Most important step!

- Identify and collect, clean and transform data
- Explore data
- Ascertain quality and ability to answer questions

May consume large proportion of the total effort

- Feature engineering
- Select model, apply and validate

Feature engineering requires deep domain knowledge

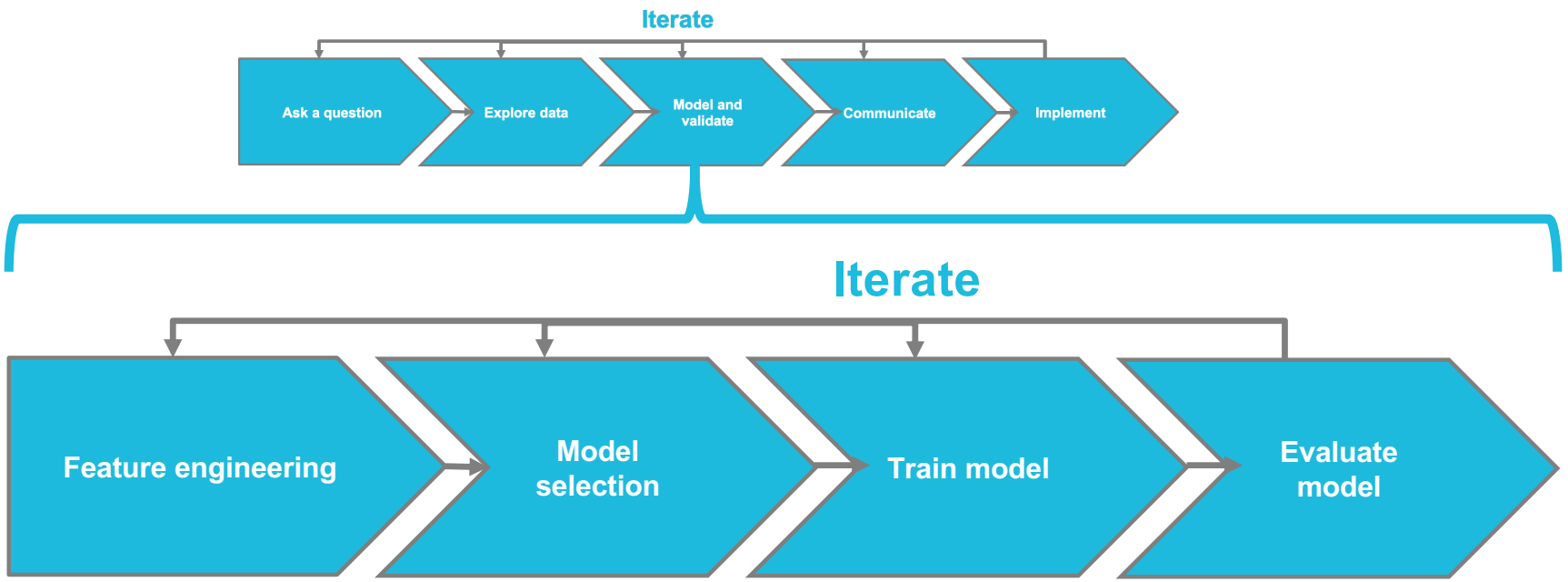
- Communicate to stakeholders and obtain buy-in

Perform as early and as often as possible

- Develop end-to-end solution
- Build, test, deploy and monitor

Consider the total cost of implementation earlier in the process

Modelling Process



- Dimensionality reduction
- Remove noise
- Feature standardisation
- Categorical feature encoding

- Select appropriate model(s)
- Select hyper-parameters

- Train model

- Accuracy, recall

Features

- A **feature** is an attribute, property or variable that is used by the Machine Learning model to perform its task such as classification. It may correspond directly to **a raw data variable**, but in most cases, a feature is created from **one or more raw data variables** using some extraction or encoding function.
- Feature engineering sit between EDA and the machine learning modeling. **Domain knowledge** and **model requirements** inform tasks in feature engineering.
- **Better features** can produce simpler and more flexible models, and they often yield better results.

Feature Engineering - Definition

- Feature engineering is the process of using **domain knowledge** and **data analysis** techniques to create features from raw data that make machine learning algorithms work effectively and efficiently.
- Feature Engineering involves the **discovery, analysis, selection, encoding** and **validation** of features before apply Machine Learning models.
- The above definition combines all Data Science activities around the features. Some sources restrict Feature Engineering to the encoding of features. They also identify others activities under different names such as feature selection and feature encoding.
- There is an emerging trend to automate some of these activities in what is referred to as **Feature Learning**.

Feature Engineering - Overview

- Feature Engineering is considered one of the **most critical step** in Data Science.
- Feature Engineering requires **domain knowledge**, **data analysis techniques** and ***creativity*** .
- Because the right features can only be defined in the context of both the model and the data; since data and models are so diverse, it's difficult to generalise the practice of feature engineering across projects.
- Also, because of the **creative aspects** of Feature Engineering, it is not a topic that **cannot be covered exhaustively**. Focus on a **domain** and continuous **practice** are the best way to develop expertise.

Feature Engineering - Overview

- Feature Engineering starts in the **Exploratory Data Analysis** where you can find data health issues, outliers, noise, feature engineering ideas, feature cleaning ideas, etc.
- Because **Simpler** models are always better! The main aim of Feature Engineering is to extract more new more useful features and **remove** irrelevant or noisy features.

Feature Engineering - process

- **Feature discovery** (starts in the **EDA** stage)
- **Feature encoding** (aka feature extraction)
- **Feature selection** (can overlap with the model evaluation)
- Identify issues or gaps and **iterate** from the top

Feature discovery

- Feature discovery starts during EDA.
- Feature discovery is simply the review of each variable to assess its relevance for the task at hand.
- You should try to cover **as many candidate features** as possible to ensure that you **do not miss** some potentially useful information.
- However, you should do feature discovery (and encoding) **incrementally** to avoid **wasting time** on features that do not contribute much to the task at hand.
- So, the best approach is to **train, test and evaluating the model** with a **handful** of features. Then **incrementally** discover and encode more features.

Feature discovery – feature analysis

- Investigate the feature and its **business (domain) significance**
- Understand how this feature was collected and assess the ability to obtain this data in the future. **Data integration** techniques may be needed here.
- Review **statistics** of the features such as mean, variance, missing data, outliers, etc using combination of **data analysis** and **visualisation** techniques.
- Examine relationship among features and between features and target.
- Sometimes features can form a **group** (for example, if they represent different aspect of the **same underlying entity**) which may provide an opportunity to eliminate some of these features or combine them.

Feature encoding (aka feature extraction)

- Feature encoding aims to **extract and prepare** features to be used effectively by the Machine Learning models.
- Some sources refer to feature encoding activity as **feature engineering** or **feature extraction**.
- There are many techniques for feature encoding depending on the **type and nature** of the features.
  - **Tabular features** (simple numeric of categorical features)
    - We will focus for the time being on these type of features.
  - Natural language text or documents
  - Speech and signals from various devices and IoT edge nodes.
  - Images and videos

Feature encoding - Tabular data encoding

- Tabular data is usually limited to **simple** numeric or categorical values.
- **Numerical data**
  - Numerical data may require feature encoding depending on the models used. The encoding may include **simple techniques** such as normalisation or more **advanced techniques** to extract information from multiple data variables.
- **Categorical data**
  - Some algorithms such decision trees **support categorical values** without encoding. Other algorithms such as Logistic Regression **do not support categorical values** and therefore necessitate the encoding of categorical data variables into numeric equivalent.

Target Variable encoding

- You can treat a **label/target/dependent** variable as a feature of the data and apply similar transformation on it.
- Target variable encoding can improve **model fit** when the target variable shows a skewed distribution.
- Use a encoding function such log(x), sqrt(x),etc can be useful. However, there is **no general rules** on which function to use.
- Avoid **leakage** which occurs when you encode information from the target into a feature.

Feature encoding - Numerical features

- Numerical data can be fed directly to Machine Learning algorithms. However, some encoding is usually required.
- Note that sometimes **high precision** is just noise. **Rounding** can be an effective simple encoding to improve the results of the models.
- **Binning** can be another useful encoding. The optimal number of bins depending on the domain and data distribution.
  - **Fixed-width** binning
  - **Quantile** binning
- Removing **outliers** can be also useful but does not apply in some use cases such as **anomaly detection**.

Feature encoding - scaling

- Scaling or normalisation is needed for many ML algorithms that compute **distances** between observations. Scaling **standardises** the spread of features.
- Scaling can be achieved with:
  - **Mean scaling**
    - $x` = \frac{x - \overline{x}}{x_{max} \_ x_{min}}$
  - **Min-max scaling**
    - $x` = \frac{x - xmin}{x_{max} \_ x_{min}}$
  - **Standardisation**
    - $x` = \frac{x - \overline{x}}{\sigma}$

Feature encoding - categorical features

- In most cases, categorical features require some **transformation** before applying Machine Learning models. For example, by using **dummy features.**
- **One-hot encoding** transforms categories into individual binary (0 or 1) features. Can be applied using scikit-learn: DictVectorizer, OneHotEncoder.
- The main difference between dummy feature and one-hot encoding is that dummy features encoding can produce **(k – 1)** features (when using drop_first = true) where one-hot encoding produces **k** features. Where k is the number of categories.
- High cardinality creates **high dimension** data and very **sparse** data that may cause problems for some models.

Feature encoding - Imputation

- Imputation is an effective way for dealing with **missing data** instead of removing rows and columns.
- Effective imputation depends on the **domain, the type** and nature of the data nd feature.
- Note that missing values such as Null or NaN may hold information. Use **domain knowledge** to decide how to deal with missing data.
- Numerical data
  - Using **mean** is the simplest way.
  - Using **median** can be more robust to outliers.
- Categorical data
  - **Missing data** in categorical features is usually **hard to impute**. Using most frequent value is possible but may **not be appropriate for some domains or use cases**. Another alternative is to use **'Other'** for missing values.

Feature encoding – Temporal variables

- Temporal variables such **date and time** require usually special feature extraction and encoding. For example, **date of birth** may be converted to **age**.
- It is usually useful to analyse and extract macro trends such as **seasonality,** or **contemporaneous weather or significant events**.
- **Cyclic features** such as day of the week or month can be tricky to model. Techniques include mapping into two coordinates on a circle could be used if applicable for the use case.
- Modelling **trends** (instead of static numbers) can be important for many use cases. For example, capturing customer's last week spend or last month spend, etc instead of total spend.

Feature encoding - Spatial features

- Spatial variables are variables that encode a **location** in space.
- Examples include: **GPS-coordinates**, cities, countries, addresses.
- **Closeness** to key landmarks (such as hospitals, schools, city centre) could be used instead of the GPS-coordinate.
- Examples:
  - Distance of, say mobile phone, to **nearby businesses** could be used to generate recommendations, etc.
  - **Distance between subsequent events** could be used to detect anomalies as in fraud detection use cases.

Feature encoding – Scikit Learn

- Pandas and Scikit Learn offer a number of functions that can be used for feature encoding.
- Examples of these functions include:
  - Pandas functions for detecting and filling missing data
  - Scikit functions in the library pre-processing data such as:
    - Standardization, or mean removal and variance scaling
    - Non-linear transformation
    - Normalisation
    - Encoding categorical features
    - Discretisation (binning)
    - Imputation of missing values
    - Generating polynomial features
    - Custom transformers
    - Imputation of missing values (sklearn.impute)

Feature selection - The curse of dimensionality

- When the number of features/dimensionality increases, the **volume of the space** increases so fast that the available data become **sparse**.
- To obtain a **statistically significant result**, the amount of data needed often grows exponentially with the dimensionality.
- Endeavour always to **minimise** the number of features in your model **without losing valuable information**.
- **Combining features** to form one feature can be very useful in reducing the number of the model feature without losing any information.
- Dimensionality reduction using techniques such as PCA can be used. Do not use dimensionality reduction techniques as **the first resort** as they tend to complicate and hinder the interpretability of the model. However, you should use PCA in EDA to visualise the data.

Feature selection

- Feature selection is one of the ways to **reduce dimensionality** to avoid overfitting by reducing the complexity of the model.
- There are many techniques that can be used feature selection. These techniques should be *always used in conjunction with* ***domain knowledge***. They are usually grouped under:
  - **Filter methods**
  - **Wrapper methods**
  - **Embedded methods.**
- Although the use of **PCA** to reduce dimensionality is not, strictly speaking, a type of feature selection, it achieves a similar outcome in an indirect way.
- **Feature learning** is an emerging techniques that is used as a part of **Deep Learning** frameworks.

Feature selection - Filtering

- Filtering methods remove individual features based on some **statistics** scores **independent of any machine learning algorithm**.
- Filtering techniques are **much cheaper** than the wrapper techniques described next but they do not take into account the **model** being employed.
- Filtering methods
  - Remove features with many **missing values.**
  - Remove features with **minimal variations.**
    - Scikit Learn **VarianceThreshold** function removes features with variance lower than a specified threshold.
  - Remove one of **correlated features** (pairwise correlation).
  - Remove features with a weak **correlation with the target.**

Feature selection – Wrapper methods

- Wrapper method uses a machine learning model and uses its performance as evaluation criteria to select features.
- Forward, backward or stepwise selection:
  - The procedure starts with either an empty set of features or all features and selects the best performing subset of features.
- Scikit Learn Univariate feature selection
  - **Univariate feature selection** selects the best features based on their contribution to the model performance given a a scoring function. You can specify the number of best features to keep or percentile.
  - **Recursive feature elimination** takes an existing model that assigns weights to features and selects features by recursively considering smaller and smaller sets of features. It uses the **model accuracy** to identify which features (and combination of features) contribute the most to predicting the target. It can be used with **cross validation**.
  - **Scikit Learn Feature selection** using SelectFromModel which can be used with any model that produces **coefficients** that indicate importance of features. Recursive feature elimination removes features with coefficients below certain threshold.

Feature selection – Embedded methods

- Embedded methods are iterative in a sense that they takes care of each iteration of the model training process and extract those features which contribute the most to **performance of the model**.
- **Regularisation methods** are the most commonly used embedded methods which penalise a feature given a coefficient threshold.
- Examples of regularisation algorithms include the **LASSO**, **Elastic Net** and **Ridge Regression**.

Feature selection - Use of clustering

- Insights from **unsupervised clustering** can be used to decide on the appropriate features for the task at hand which uses supervised learning techniques such as classification.
- Clustering can be used during Feature Engineering to discover **relationships and trends** in the features space.
- Clustering techniques can be used for feature selection by using the **membership of a cluster** as a feature.
- **K-Means** is the most common clustering algorithm.

“...some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”

Pedro Domingos

Questions?

Appendices

Dummy Variables

How can we use categorical variables in an algorithm that requires numerical predictors?

- ***ordinal categoricals***
  - can be converted to a sequence of integers, if it makes sense to do so

cold	cool	moderate	warm	hot
1	2	3	4	5

  - the above implies hot < warm < moderate < cool < cold
    ... which makes sense

Dummy Variables

How can we use categorical variables in an algorithm that requires numerical predictors?

- ***cardinal categoricals***

apples	bananas	peaches	oranges	pears
1	2	3	4	5

> *this implies pears > oranges > peaches > bananas > apples ... does not make sense!*

> must convert to dummy variables instead

Cardinal Dummy Variables

- full definition (number of variables = number of categories):
  - fruit_apples: 1 = apples, 0 = no apples
  - fruit_bananas: 1 = bananas, 0 = no bananas
  - fruit_peaches: 1 = peaches, 0 = no peaches
  - fruit_oranges; 1 = oranges, 0 = no oranges
  - fruit_pears; 1 = pears, 0 = no pears

- compact definition (number of variables is one less than number of categories):
  - fruit_bananas: 1 = bananas, 0 = apples
  - fruit_peaches: 1 = peaches, 0 = no peaches
  - fruit_oranges; 1 = oranges, 0 = no oranges
  - fruit_pears; 1 = pears, 0 = no pears

Python:

pandas.get_dummies

End of presentation