Data Engineering and ML Pipelines with Python

Delivered by: Noureddin Sadawi, PhD

About the Speaker

Name: Dr Noureddin Sadawi

Academic Qualification: PhD Computer Science

Experience: Several areas including but not limited to Docker, Machine Learning and Data Science, Python and more.

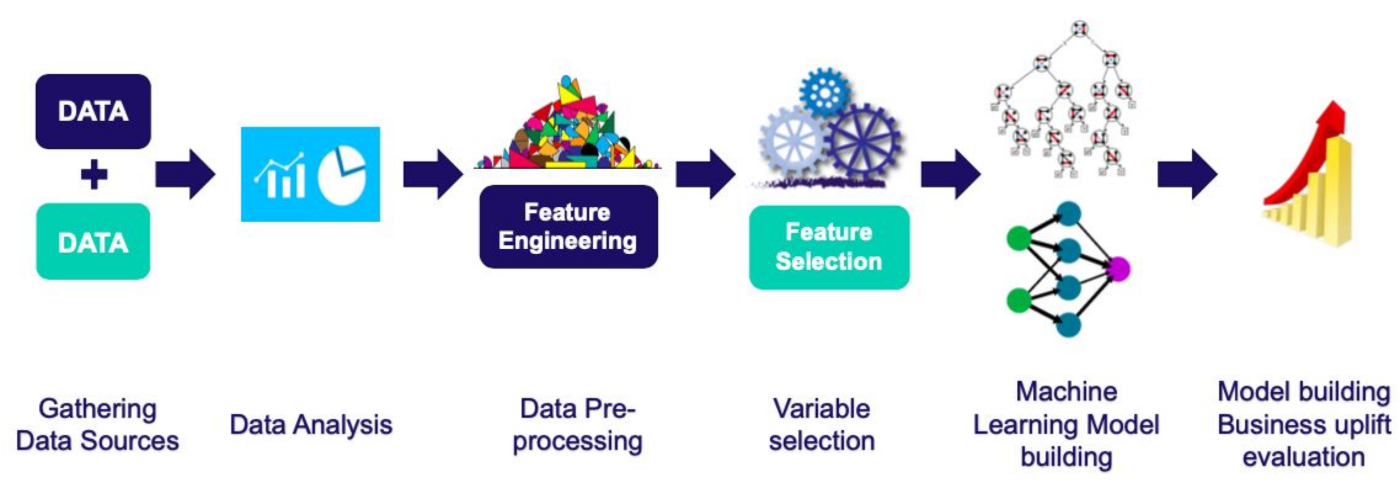
aws CERTIFIED

Machine Learning Specialty

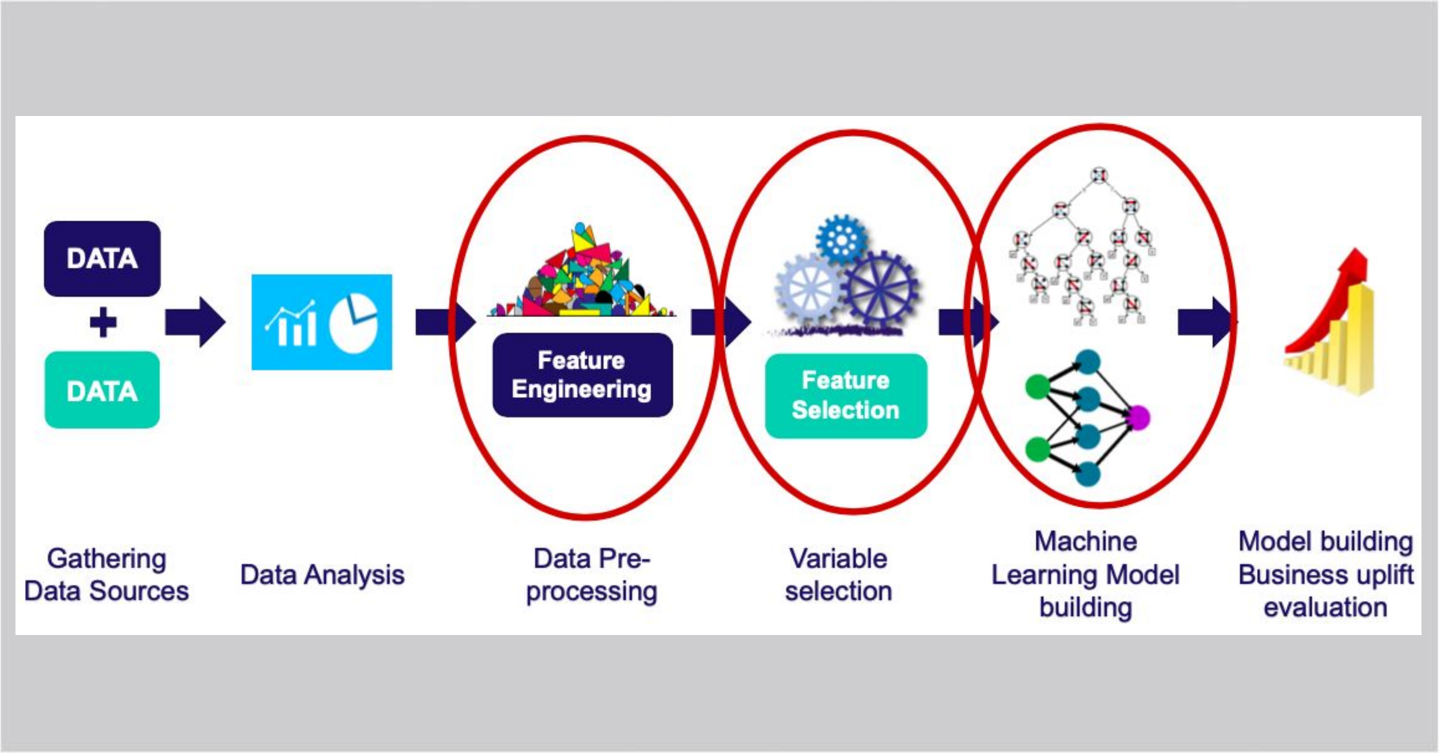
UNIVERSITY OF OXFORD

UNIVERSITY OF LONDON

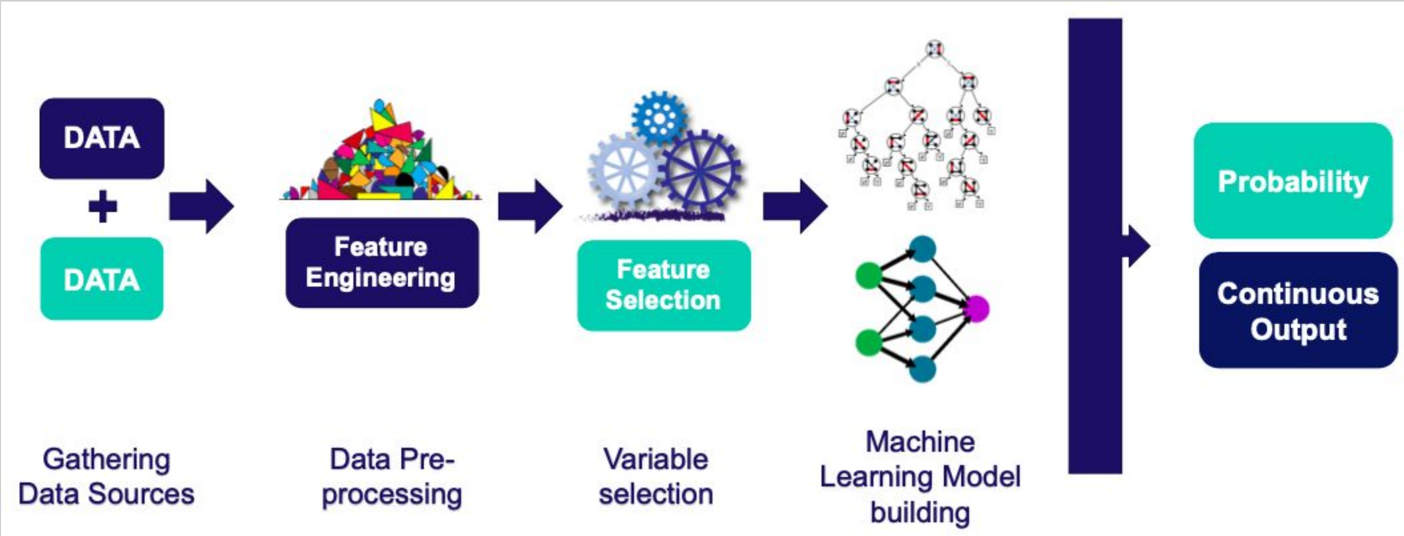
Machine Learning Pipeline: Overview



Machine Learning Pipeline: Production



Machine Learning Pipeline: Production



Feature Engineering

- Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used in machine learning.
- In order to make machine learning work well on new tasks, it might be necessary to design and train better features.
- A machine learning technique that leverages data to create new variables that aren't in the training set.

Feature Engineering

- It can produce new features for both supervised and unsupervised learning, with the goal of **simplifying and speeding up data transformations** while also **enhancing model accuracy**.
- Feature engineering is required when working with machine learning models.
- Regardless of the data or architecture, a terrible feature will have a direct impact on your model.

Feature Selection

- The process of reducing the number of input variables when developing a predictive model.
- It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.
- Select a subset of original features, not replace them with new ones (compare with dimensionality reduction).

Why Feature Selection?

- Simple models are easier to interpret.
- Shorter training times.
- Enhanced generalisation by reducing overfitting.
- Easier to implement by SW developers -> Model production (i.e. less code to process and handle features).
- Reduced risk of data errors during model use.
- Data redundancy (many features provide similar info).

Feature Preprocessing

- The technique of making raw data into more meaningful data or the data which can be understood by the Machine Learning Model.
- Real-world **data** is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.
- To tackle this, data preprocessing technique is introduced.
- Example techniques :
    - Vectorization (e.g. one-hot encoding).
    - Normalization (i.e. Scaling) and Standardization.
    - Handling Missing Values.

Missing Data

- A very common problem in the real-world.
- Missing data can be caused randomly or systematically. https://www.theanalysisfactor.com/missing-data-mechanism/
- This problem affects machine learning models.
- Several methods to deal with it.

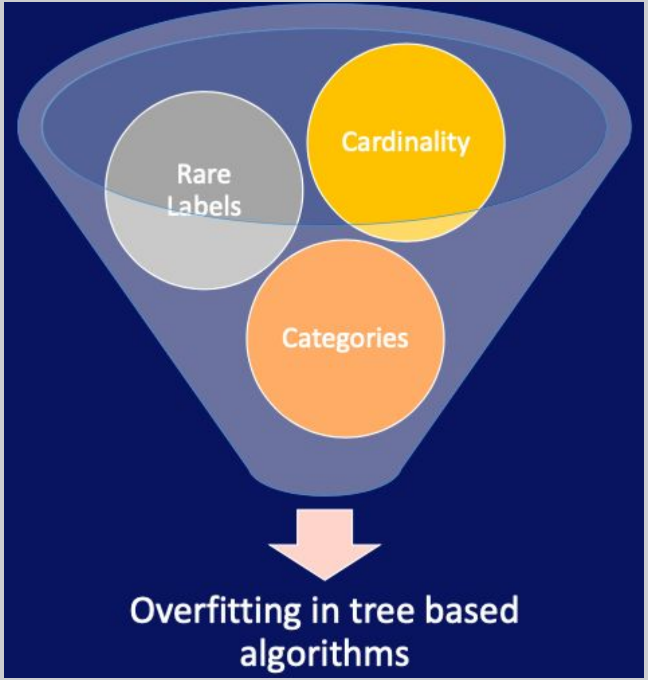
Missing Data Imputation Techniques

Numerical Variables	Categorical Variables	Both
Mean / Median Imputation	Frequent category imputation	Complete Case Analysis
Arbitrary value imputation	Adding a “missing” category	Adding a “Missing” indicator
End of tail imputation		Random sample imputation

- Scikit-learn: https://scikit-learn.org/stable/modules/impute.html
- Book: https://stefvanbuuren.name/fimd/

Labels in Categorical Variables

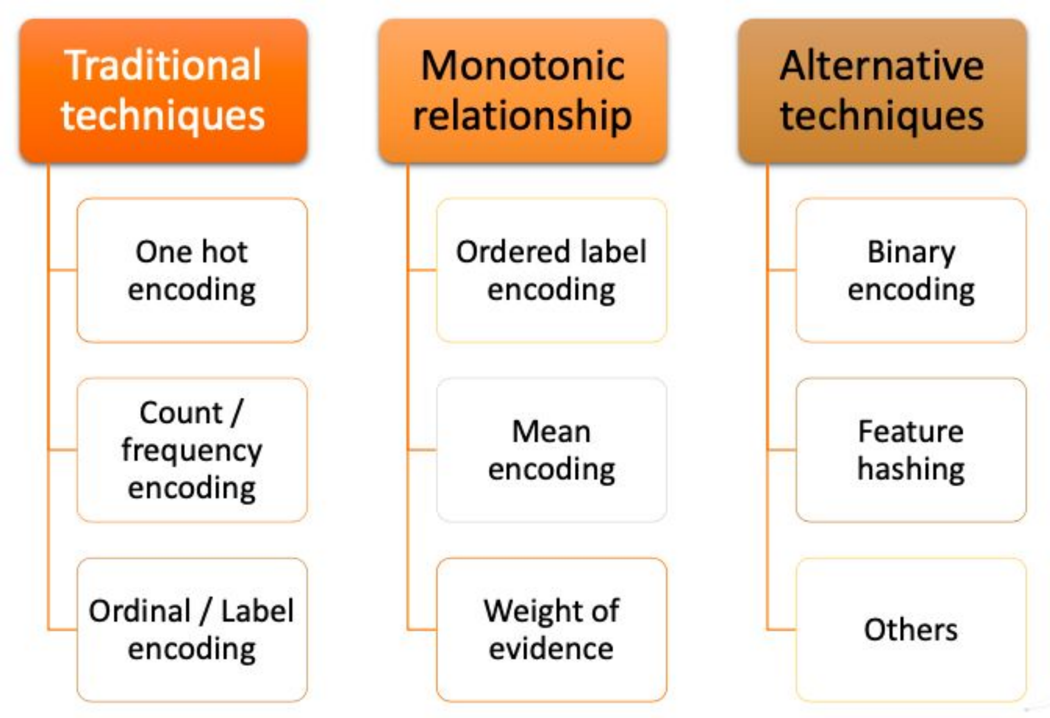
- Cardinality: high number of labels.
- Rare labels: infrequent categories.
- Categories: strings.
- Tree-based methods tend to overfit to variables with high cardinality.
- Useful tutorials here, here and here.



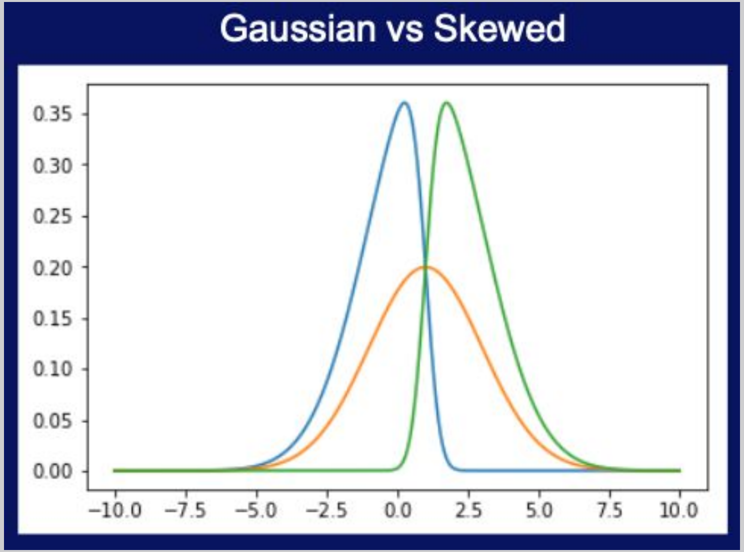
Rare Labels

- Really important for model deployment.
- Group rare labels into one group.
- One-hot encoding of frequent categories.

Categorical Encoding Techniques



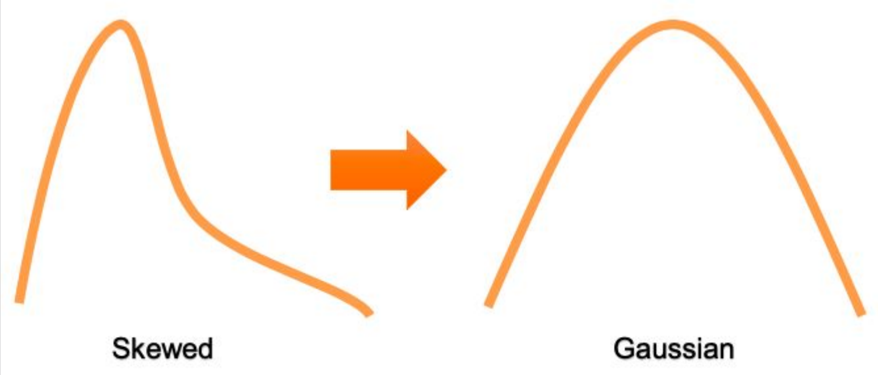
Distributions



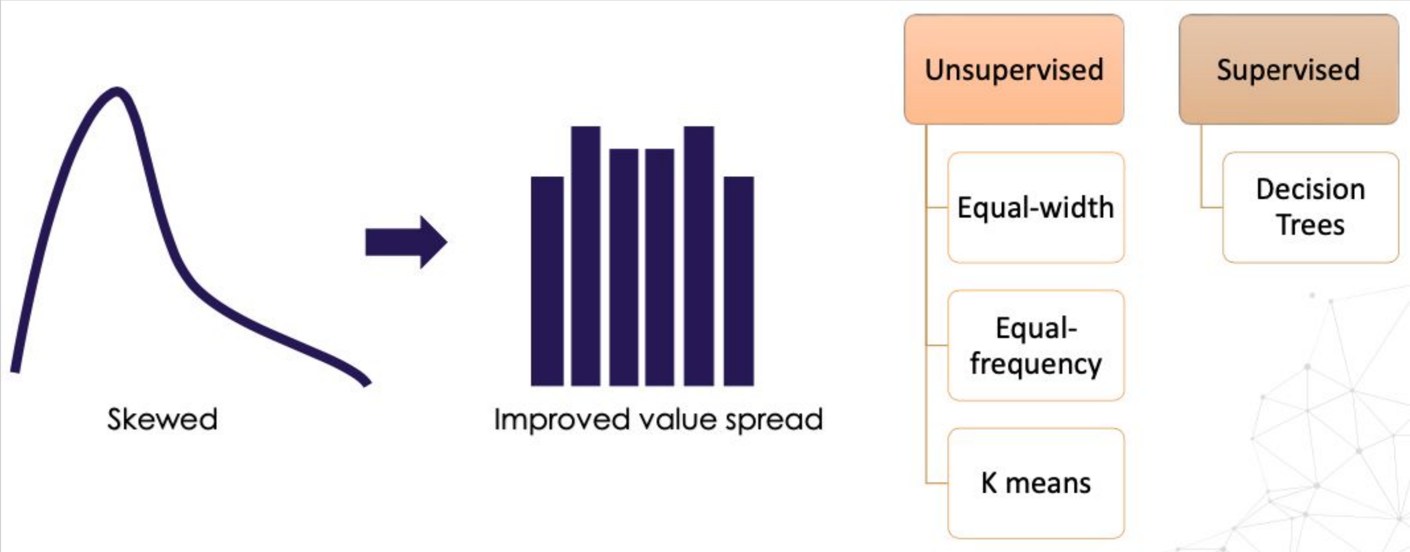
- Better spread of values may benefit performance.
- Some models make assumptions on the variable distributions.

Mathematical Transformation

- Logarithmic.
- Exponential.
- Reciprocal.
- Box-Cox.
- Yeo-Johnson.



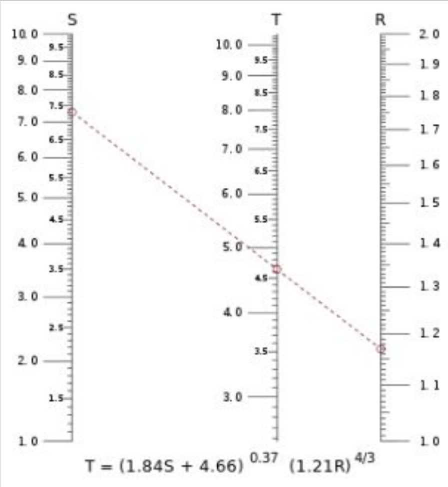
Discretization



Feature Magnitude - Scale

ML models sensitive to feature scale:

- Linear and Logistic Regression
- Neural Networks
- Support Vector Machines
- KNN
- K-means clustering
- Linear Discriminant Analysis (LDA)
- Principal Component Analysis (PCA)

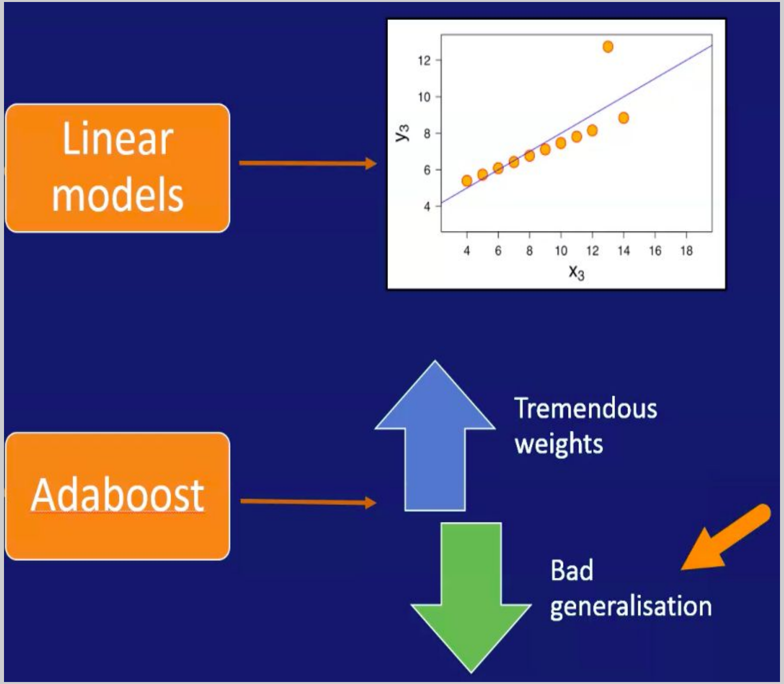


Tree based ML models insensitive to feature scale:

- Classification and Regression Trees
- Random Forests
- Gradient Boosted Trees

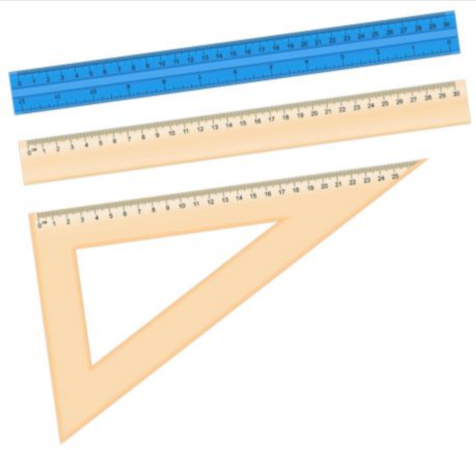
Outliers

- Outliers are values that are extremely low or extremely high compared with the remaining values of a variable.
- They can affect some ML models.
- How to deal with them?
  - Capping.
  - Truncation/Censoring.
  - Discretisation.



Feature Scaling Methods

- Standardisation.
- Mean Normalisation.
- Scaling to Max and Min.
- Scaling to Absolute Max.
- Scaling to Median and Quantiles.
- Scaling to Unit Norm.



Nice Table: https://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-7-142/tables/1

Datetime Data

- Day, Month, Year.
- Hour, Minute, Second.
- Elapsed Time:
  - Time between transactions.
  - Age.

Timeseries and Transactions

Aggregate data

- Number of payments in last 3, 6, 12 months
- Time since last transaction
- Total spending in last month

Text Data

- Characters, words, unique words.
- Lexical diversity.
- Sentences, paragraphs.
- Bag of Words.
- TF-IDF.
- Embeddings.


Mane joined Liverpool for **£34m from South**
games, finishing last season with 23 goals in

"Sadio Mane is a world star who underscores
attractiveness of the entire Bundesliga," said
unique footballers that the fans come to the

Geo Data

- Distances.

Feature Combination

- **Ratio:** Total debt with income -> Debt to income ratio
- **Sum:** Debt in different credit cards -> total debt
- **Subtraction:** Income without expenses -> disposable income

Reproducibility 1/3

- The ability to replicate a model precisely, so that given the exact same raw data as an input, the reproduced model will return the same output.
- Lack of reproducibility can have numerous negative effects.
- From a financial standpoint, if one were to invest significant resources into creating a model in a research environment, but they were unable to reproduce it in a production environment, little benefit would come of that model and its predictions.

Reproducibility 2/3

- Similarly, this would result in wasted time and effort, in addition to the financial loss.
- The machine learning model serves little use outside of the research environment.
- Most importantly, however, is that reproducibility is crucial to replicating prior results. Without this, one could never accurately determine if new models exceeded previous ones.

Reproducibility 3/3

- Slides by Rachael Tatman: https://www.rctatman.com/files/Tatman_2018_ReproducibleML.pdf
- Nice article by Sole Galli: https://trainindata.medium.com/how-to-build-and-deploy-a-reproducible-machine-learning-pipeline-20119c0ab941
- The Machine Learning Reproducibility Crisis by Pete Warden: https://petewarden.com/2018/03/19/the-machine-learning-reproducibility-crisis/
- Building a Reproducible Machine Learning Pipeline: https://arxiv.org/ftp/arxiv/papers/1810/1810.04570.pdf

Reproducibility with Keras and NNs

- How to Get Reproducible Results with Keras: https://machinelearningmastery.com/reproducible-results-neural-networks-keras/
- Reproducibility in ML: why it matters and how to achieve it: https://www.determined.ai/blog/reproducibility-in-ml
- StackOverflow: https://stackoverflow.com/questions/32419510/how-to-get-reproducible-results-in-keras

Research vs Production Environments

	Research	Production
Separate from customer facing software	✓	x
Reproducibility matters	Sometimes	Almost always
Scaling challenges	x	✓
Can be taken offline	✓	x
Infrastructure planning required	Sometimes	Almost always
Difficult to run experiments	x	✓

Data Exploration Example

Python Code on Jupyter Notebook

House-price-data-exploration

Feature Engineering Code Example

Python Code on Jupyter Notebook

Predicting-Titanic-survival-manual

End of Part 1

Open Source for Feature Engineering

- Scikit-learn: https://scikit-learn.org/
- Feature Engine: https://feature-engine.readthedocs.io/
- Category Encoders: https://contrib.scikit-learn.org/category_encoders/
- Featuretools: https://www.featuretools.com/
- Imbalanced-learn: https://imbalanced-learn.org/

Open Source for Feature Selection

- Scikit-learn: https://scikit-learn.org/
- Feature Engine: https://feature-engine.readthedocs.io/
- MLXtend: http://rasbt.github.io/mlxtend/

Open Source for Model Training

- Scikit-learn: https://scikit-learn.org/
- MLXtend: http://rasbt.github.io/mlxtend/
- Tensorflow: https://www.tensorflow.org/
- Pytorch: https://pytorch.org/
- Keras: https://keras.io/

Many others!

Scikit-Learn

- Solid implementation of a wide range of ML algorithms and data transformations.
- Clean, uniform, and streamlined API.
- Most algorithms follow the same functionality -> implementing new algos is really easy:
    - Transformers.
    - Estimators.
    - Pipeline.
- Complete online documentation, with some theory & examples.
- Well established in the community -> new packages follow Scikit-learn functionality to be quickly adopted by end users, e.g., Keras, MLXtend, category-encoders, Feature-engine.

Data Transformers

- Scikit-learn: https://scikit-learn.org/stable/data_transforms.html
- Feature-engine: https://feature-engine.readthedocs.io/en/latest/#feature-engine-s-transformers
- Category encoders: https://contrib.scikit-learn.org/category_encoders/

Scikit-Learn Estimators

- **Estimator** - a class with *fit()* and *predict()* methods.
- It fits and predicts.
- All ML algorithms are coded as estimators within Scikit-Learn.

```python
class Estimator(object):

    def fit(self, X, y=None):
        """
        Fits the estimator to data.
        """
        return self

    def predict(self, X):
        """
        Compute the predictions
        """
        return predictions
```

Scikit-Learn Transformers

- **Transformer** - a class with *fit()* and *transform()* methods.
- It transforms data.
    - Scalers.
    - Feature selectors.
    - Encoders.
    - Imputers.
    - Discretizers.
    - Transformers.

```
class Transformer(object):

    def fit(self, X, y=None):
        """
        Learn the parameters to
        engineer the features
        """

    def transform(X):
        """
        Transforms the input data
        """
        return X_transformed
```

Scikit-Learn Pipelines

- **Pipeline** - a class that allows to run transformers and estimators in sequence.
- Most steps are Transformers.
- Final step can be an Estimator.

```
class Pipeline(Transformer):

    @property
    def name_steps(self):
        """Sequence of transformers
        """
        return self.steps

    @property
    def _final_estimator(self):
        """
        Estimator
        """
        return self.steps[-1]
```

Pipeline

Category Encoders

```
# train pipeline
price_pipe.fit(X_train, y_train)

# transform data
price_pipe.predict(X_train)
price_pipe.predict(X_test)


price_pipe.predict(live_data)
```

Scikit-Learn Pipeline in Action

```
vect = CountVectorizer()
tfidf = TfidfTransformer()
clf = SGDClassifier()

vX = vect.fit_transform(Xtrain)
tfidfX = tfidf.fit_transform(vX)
predicted = clf.fit_predict(tfidfX)

# Now evaluate all steps on test set
vX = vect.fit_transform(Xtest)
tfidfX = tfidf.fit_transform(vX)
predicted = clf.fit_predict(tfidfX)
```

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', SGDClassifier()),
])
predicted = pipeline.fit(Xtrain).predict(Xtrain)
# Now evaluate all steps on test set
predicted = pipeline.predict(Xtest)
```

- Nice answer on SO: https://stackoverflow.com/a/33094099
- Example on Scikit-Learn website.

Good Pipeline Code Examples

https://pythonguides.com/scikit-learn-pipeline/

Pipeline with Feature-engine:
https://feature-engine.readthedocs.io/en/latest/quickstart/index.html

Procedural Programming in ML

Code:

- Learn the parameters.
- Make the transformations.
- Make the predictions.

Data:

- Store the parameters.
- Mean values, regression coefficients, etc.

Object Oriented Programming (OOP)

- In Object-oriented programming (OOP) we write code in the form of “objects”.
- An object can store data and can also store instructions or procedures (code) to modify that data, or do something else, like obtaining predictions.

- Data ⇒ attributes, properties.
- Code or Instructions ⇒ methods (procedures).

OOP for ML

In Object-oriented programming (OOP) the "objects" can learn and store parameters.

- Parameters get automatically refreshed every time model is re-trained.
- No need of manual hard-coding.

- Methods:
  - Fit: learns parameters.
  - Transform: transforms data with the learned parameters.
- Attributes: store the learnt parameters.

Class

- The properties or parameters that the class takes whenever it is initialized, are indicated in the __init__() method.
- Methods are functions defined inside a class and can only be called from an instance of that class.
- Our fit() methods learns parameters and our transform() method transforms data.

```
class MeanImputer:
    def __init__(self, variables):
        self.variables = variables

    def fit(self, X, y=None):
        self.imputer_dict_ =
        X[self.variables].mean().to_dict()
        return self

    def transform(self, X):
        for x in self.variables:
            X[x] = X[x].fillna(
                  self.imputer_dict_[x])
        return X
```

Inheritance

Inheritance is the process by which one class takes on the **attributes** and **methods** of another.

- The properties or parameters that the class takes whenever it is initialized, are indicated in the **__init__()** method.
- The first parameters will always be a variable called ***self***.
- We can give any number of parameters to __init__()

```
class TransformerMixin:

    def fit_transform(self, X, y=None):
        X = self.fit(X, y).transform(X)
        return X
```

Our MeanImputer

Inherits the method fit_transform() from the TransformerMixin.

```
>> my_imputer = MeanImputer(
>>         variables = [‘age’, ‘fare’]
>> )

>> data_t = my_imputer.fit_transform(my_data)
>> data_t.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape
0	0.083333	0.0	0.495064	0.0	0.0	0.0	0.666667
1	0.083333	0.0	0.499662	0.0	0.0	0.0	0.666667
2	0.083333	0.0	0.466207	0.0	0.0	0.0	0.666667
3	0.083333	0.0	0.485693	0.0	0.0	0.0	0.666667
4	0.083333	0.0	0.265271	0.0	0.0	0.0	0.666667

```
class MeanImputer(TransformerMixin):
    def __init__(self, variables):
        self.variables = variables

    def fit(self, X, y=None):
        self.imputer_dict_ =
        X[self.variables].mean().to_dict()
        return self

    def transform(self, X):
        for x in self.variables:
            X[x] = X[x].fillna(
                    self.imputer_dict[x])
        return X
```

Scikit-Learn API Documentation

- https://scikit-learn.org/stable/modules/classes.html
- base.BaseEstimator
- base.TransformerMixin

Important to know, when you write your own Transformer:

- Inherit from BaseEstimator to define the paremeters.
- TransformerMixin to inherit the fit_transform functionality.

Feature Engineering Pipeline Example

Python Code on Jupyter Notebook

Predicting-Titanic-survival-pipeline

End of Part 2

Scikit-Learn’s Pipeline

- Python’s sklearn package provides a “Pipeline of transforms with a final estimator”.
- The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.
- Sequentially apply a list of transforms and a final estimator.
- Intermediate steps of the pipeline must be ‘transforms’, that is, they must implement fit and transform methods.
- The final estimator only needs to implement fit.

Why Pipeline?

- Useful .. contains links to previous lessons https://medium.com/@haataa/nlp-pipeline-101-with-basic-code-example-modeling-40c75d963984
- https://towardsdatascience.com/streamlining-feature-engineering-pipelines-with-feature-engine-e781d551f470
- https://www.seldon.io/what-is-a-machine-learning-pipeline
- https://valohai.com/machine-learning-pipeline/
- https://www.iguazio.com/glossary/machine-learning-pipeline/

Benefits of ML Pipeline

- Mapping a complex process which includes input from different specialisms, providing a holistic look at the whole sequence of steps.
- Focusing on specific steps in the sequence in isolation, allowing the optimisation or automation of individual stages.
- The first step in transforming a manual process of machine learning development to an automated sequence.
- Providing a blueprint for other machine learning models, with each step in the sequence able to be refined and changed depending on the use case.
- Solutions are available for the orchestration of machine learning pipelines, to improve efficiency and automate the steps.
- Easily scalable, upscaling modular parts of the machine learning pipeline when needed.

Pipeline for grid-search

- A grid-search example to demonstrate why pipelines are better than separate steps (why pipeline is better than manual): https://towardsdatascience.com/scikit-learn-pipeline-tutorial-with-parameter-tuning-and-cross-validation-e5b8280c01fb
- Optimizing with sklearn's GridSearchCV and Pipeline (Nice code example): https://donernesto.github.io/blog/optimizing-with-sklearns-gridsearchcv-and-pipeline/
- Another code example: https://medium.com/analytics-vidhya/ml-pipelines-using-scikit-learn-and-gridsearchcv-fe605a7f9e05

Feature Selection inside the Pipeline

- Model Built and Refreshed on the Same Data:
  - If the dataset remains relatively stable in terms of features and distribution, feature selection inside the pipeline ensures consistency.
- Model Built and Refreshed on Smaller Datasets:
  - If your dataset is small, feature selection inside the pipeline helps maintain generalization without overfitting to noise.
  - It ensures that feature selection is done dynamically during each training cycle, making it adaptable.

Feature Selection not inside the Pipeline

If the Model is Built Using Datasets with a High Feature Space:

- If the number of features is very large (e.g., thousands of dimensions), feature selection inside the pipeline can become computationally expensive because it runs at every cross-validation fold.

Alternative:

- If feature selection is computationally intensive (e.g., recursive feature elimination, LASSO), do it before the pipeline to avoid redundant calculations.
- If using lightweight feature selection (e.g., SelectKBest, VarianceThreshold), it's fine to include in the pipeline.

Feature Selection not inside the Pipeline

If the Model is Constantly Enriched with New Data Sources:

- If new features are frequently added, feature selection inside the pipeline may lead to **inconsistent feature sets** across different model updates.
- **Issue:** If the selected features change over time, older trained models may not align with newer feature spaces, making the system harder to maintain.
- **Alternative:** Perform feature selection outside the pipeline and maintain a static, well-defined set of features for all future model retraining.

Model/Pipeline Persistence

joblib.dump() and **joblib.load()** provide a replacement for pickle to work efficiently on arbitrary Python objects containing large data, in particular large numpy arrays.

https://joblib.readthedocs.io/en/latest/persistence.html

Additional Useful Resources

- Hidden Technical Debt in Machine Learning Systems. Here.
- The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction (Google). Here.
- "Software Engineering for Machine Learning: A Case Study" (2019) Amershi et al. (Microsoft). Here.

Feature Engineering Pipeline Example

End-to-end Pipeline

Python Code on Jupyter Notebook


thank you
danke
謝謝
рахмат
ngiyabonga
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
tack
spas
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
mahalo
tapadh leat
kiitos
dankie
nanni
nandri
dhanyavad
maururu
gracias
хвала
asante
manana
obrigada
murakoze
tenki
bedankt
bayarlalaa
gràçie
hvala
köszönöm
ačiū
dankon
akun
djiere dieuf
tau
mochchakkeram
дякую
mamnun
enkosi
dziękuję
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
chokrane
taiku
sobodi
děkuji
sukriya
kop khun krap
obrigado
mèsi
didi madloba
sagolun
ありがとう
arigatô
takk
dakujem
trugarez
najis tuke
tanemirt
grazie
terima kasih
rahmet
kam sah hamnida
rahmat
diolch
dhanyavadagalu
shukriya
mercê
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ
merci