Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

Amazon Web Services

MLOps with AWS

Masterclass

aws

Machine Learning Operations with AWS

Day -12

Amazon Web Services

MLOps with AWS

Masterclass

aws

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED

Train_Test_Split

- In machine learning, the train-test split is a technique used to evaluate the performance of a predictive model.

The basic idea is to split the available data into two parts:

- one part is used to train the model, and the other part is used to test its performance.

Train_Test_Split

- In machine learning, the train-test split is a technique used to evaluate the performance of a predictive model.

The basic idea is to split the available data into two parts:

- one part is used to train the model, and the other part is used to test its performance.

Train_Test_Split Parameters

Parameters:

arrays : *sequence of indexables with same length / shape[0]

Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

test_size : *float or int, default=None*

If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples. If None, the value is set to the complement of the train size. If `train_size` is also None, it will be set to 0.25.

train_size : *float or int, default=None*

If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the train split. If int, represents the absolute number of train samples. If None, the value is automatically set to the complement of the test size.

random_state : *int, RandomState instance or None, default=None*

Controls the shuffling applied to the data before applying the split. Pass an int for reproducible output across multiple function calls. See Glossary.

shuffle : *bool, default=True*

Whether or not to shuffle the data before splitting. If shuffle=False then stratify must be None.

stratify : *array-like, default=None*

If not None, data is split in a stratified fashion, using this as the class labels. Read more in the User Guide.

Feature Selection

- It is 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 modelling and also in some case to improve the performance of the model and get rid of noise.

Feature Selection Methods

- Chi-square test
- ANOVA F-value

Chi-Squared Test

```
from sklearn.feature_selection import SelectKBest, chi2

selector = SelectKBest(chi2, k=2)

selector.fit(x,y)

x.columns[selector.get_support()]
```

Anova F Test

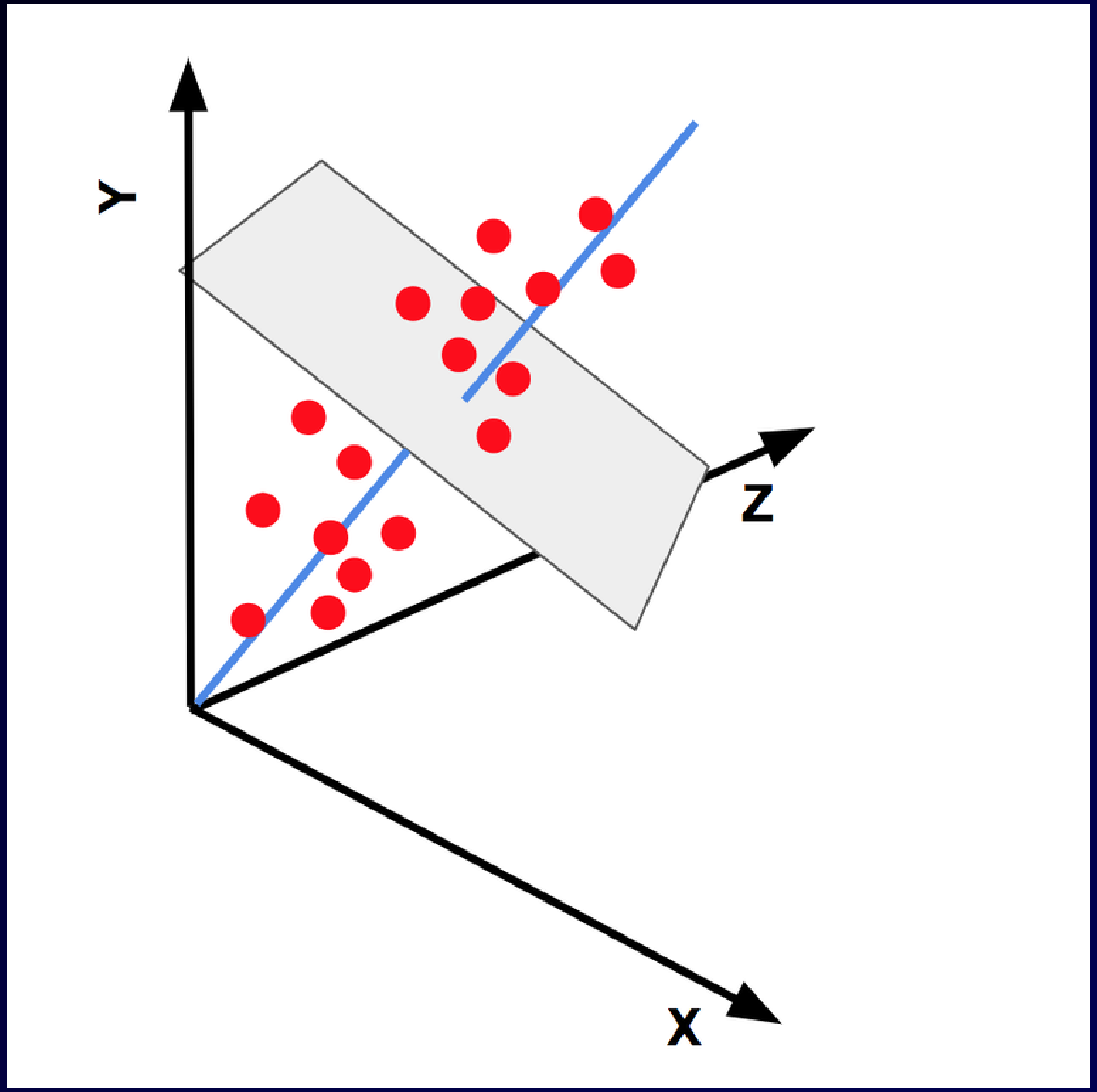
```
from sklearn.feature_selection import SelectKBest, f_classif

selector = SelectKBest(f_classif, k=2)

selector.fit(x,y)

x.columns[selector.get_support()]
```

Feature Extraction



Feature Extraction - PCA

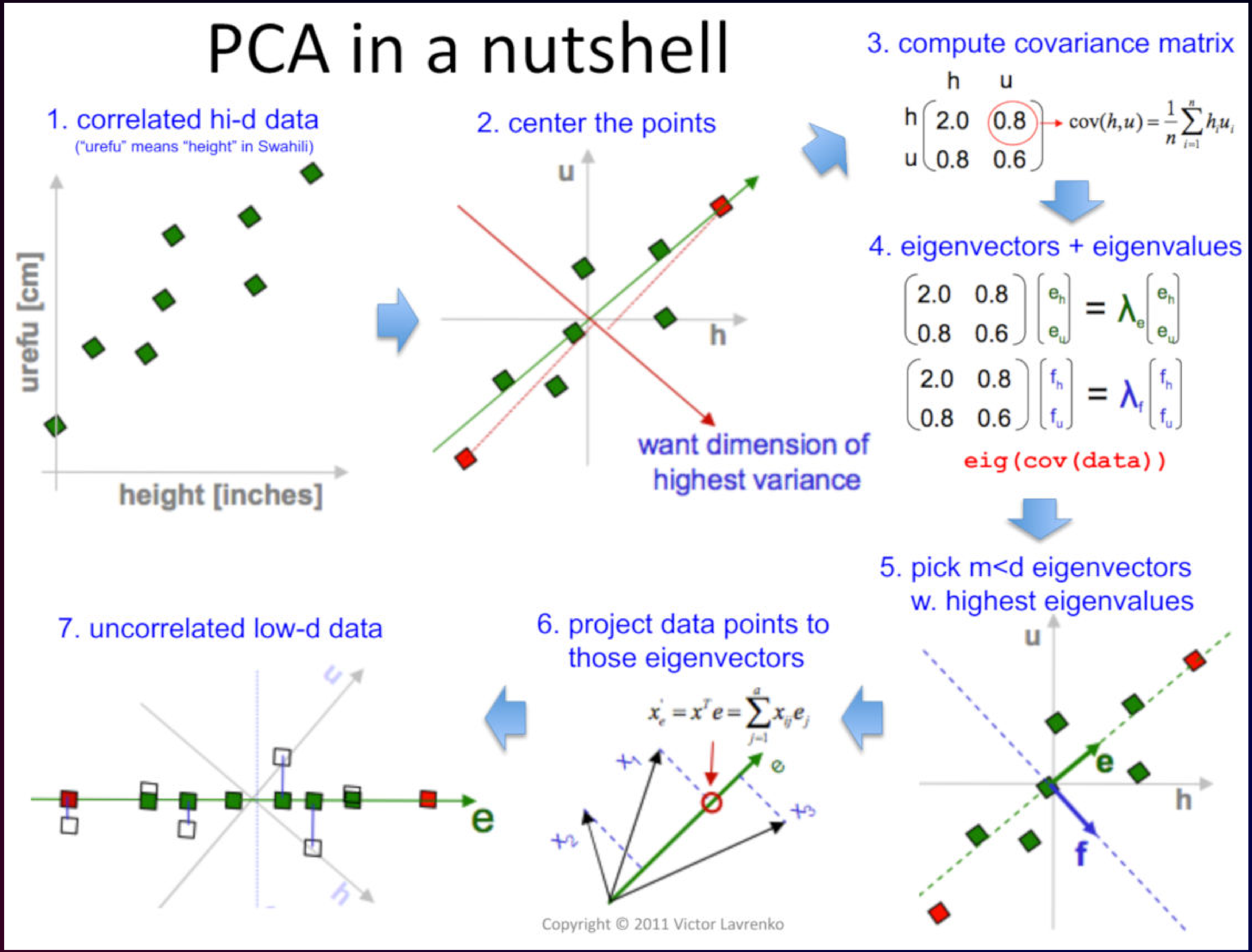
- Principal component analysis is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

- This is also called as feature extraction

Feature Selection Vs Feature Extraction

- Feature selection involves selecting a subset of the original features from the dataset for training the model. It Does not change the original features, but only selects a subset of them for model training.

- Feature extraction Involves creating new features from the original features in the dataset, which are then used for model training. It Helps to capture the underlying patterns in the data that may not be directly observable in the original feature space.


PCA in a nutshell
1. correlated hi-d data
("urefu" means "height" in Swahili)
urefu [cm]
height [inches]
2. center the points
u
h
want dimension of highest variance
3. compute covariance matrix
h u
h 2.0 0.8
u 0.8 0.6
cov(h,u) = 1/n Σ h_i u_i
4. eigenvectors + eigenvalues
2.0 0.8 0.8 0.6 e_h e_u = λ_e e_h e_u
2.0 0.8 0.8 0.6 f_h f_u = λ_f f_h f_u
eig(cov(data))
5. pick m<d eigenvectors
w. highest eigenvalues
u
e
h
f
6. project data points to
those eigenvectors
x'_e = x^T e = Σ x_ij e_j
7. uncorrelated low-d data
e
Copyright © 2011 Victor Lavrenko

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

pca.fit(data)

transformed = pca.transform(data)
```

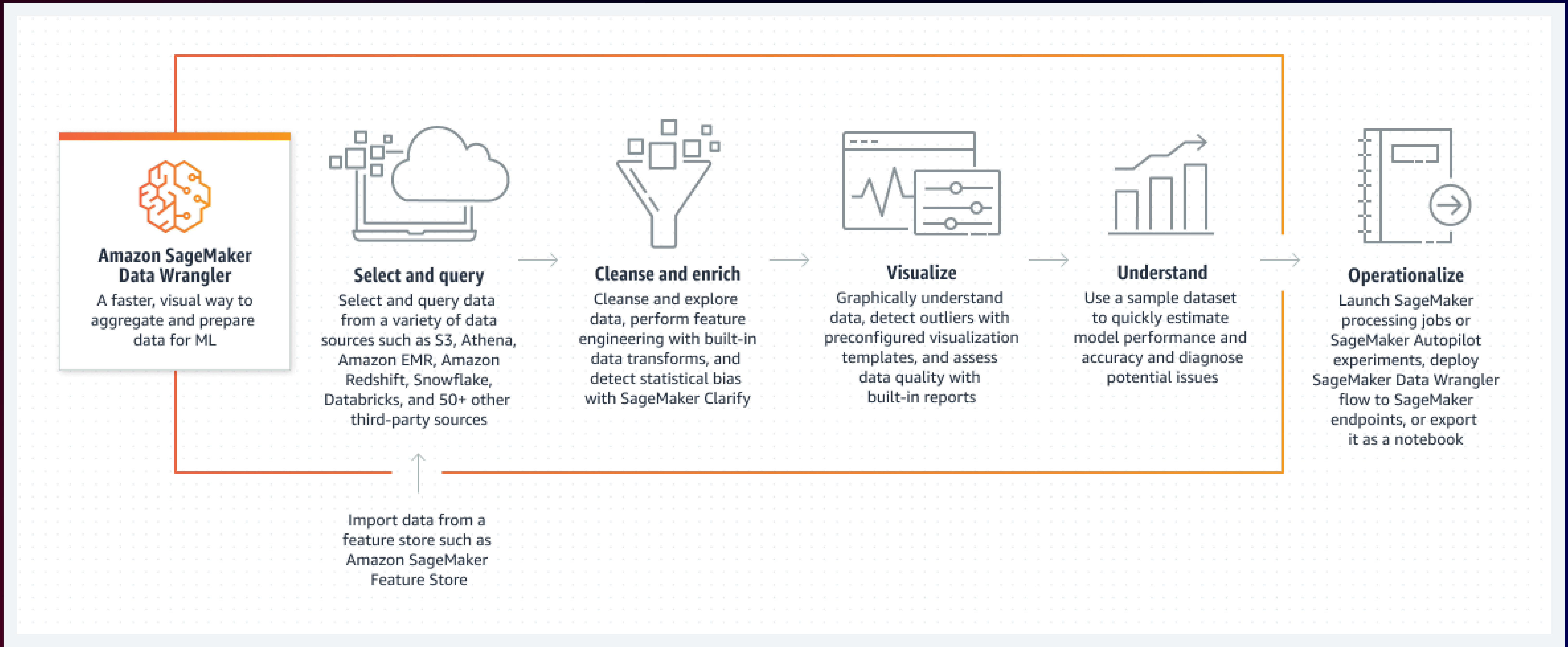
AWS Data Wrangler

AWS Data Wrangler

- SageMaker Data Wrangler, can simplify the process of data preparation and feature engineering, including data selection, cleansing, exploration, and visualization from a single visual interface.

- SageMaker Data Wrangler contains over 300 built-in data transformations so you can quickly transform data without writing any code.

AWS Data Wrangler



Thank you

Pantech e Learning
DIGITAL LEARNING SIMPLIFIED