# Move from Pandas to Spark

Adaptation of machine learning models to work in a distributed environment

# ABOUT ME…

## ANDREI GAVRILOV

Software Engineer in EPAM Systems

@ Andrei_Gavrilov@epam.com

@tbont

# Move from Pandas to Spark

## Pandas??

# Move from Pandas to Spark

## Pandas??

## Spark??

# Data Science Project WorkFlow

**Idea**

# Data Science Project WorkFlow

**Idea**

**DS team**

# Data Science Project WorkFlow

**Idea**

**DS team**

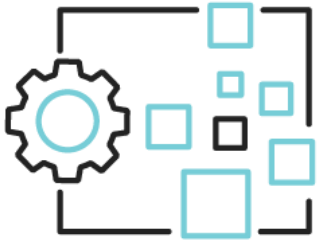
**Research**

# Data Science Project WorkFlow

**Idea**

**DS team**

**Research**



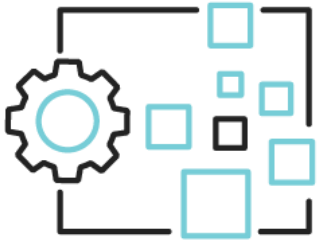
**Concept**

# Data Science Project WorkFlow

Idea

DS team

Research



Concept

PoC

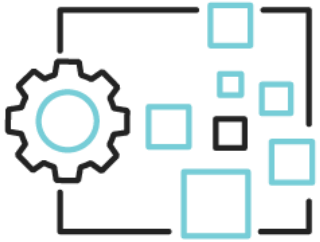
Feedback

# Data Science Project WorkFlow

Idea

DS team

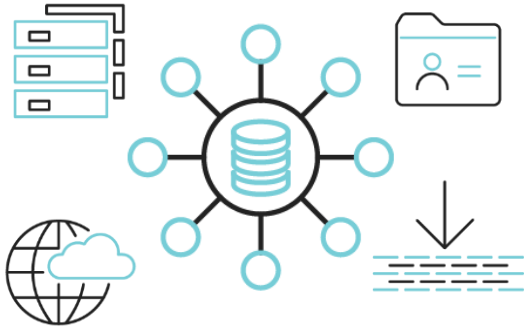
Research



Concept

PoC

Customer's Data



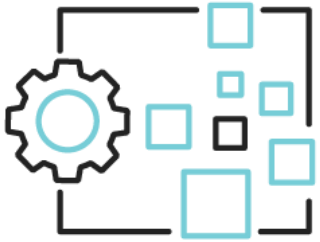
Feedback

# Data Science Project WorkFlow

Idea

DS team

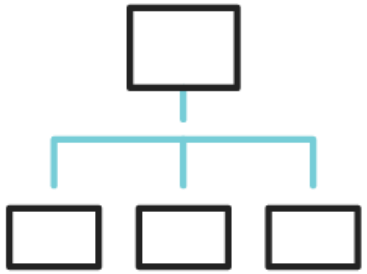
Research



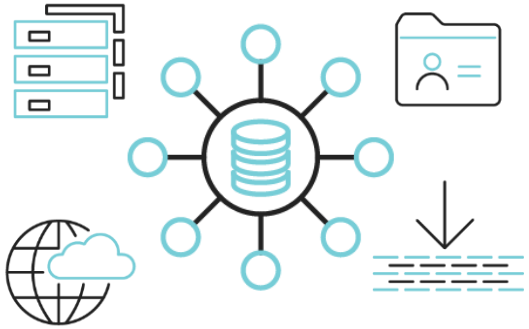
Concept

PoC

Architecture



Customer's Data



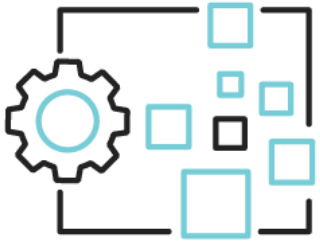
Feedback

# Data Science Project WorkFlow

Idea

DS team

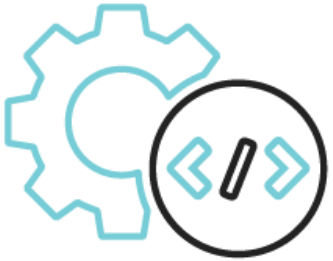
Research



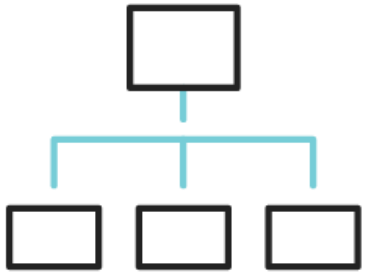
Concept

PoC

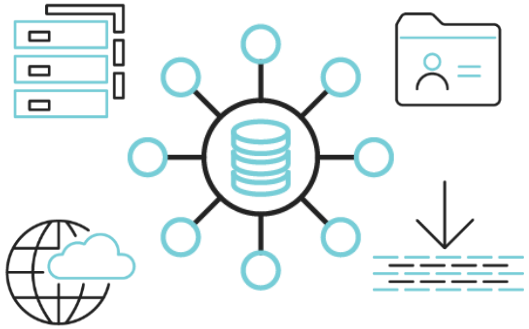
Development



Architecture



Customer's Data



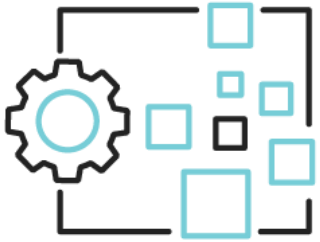
Feedback

# Data Science Project WorkFlow

Idea

DS team

Research

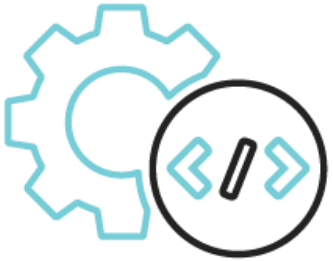


Concept

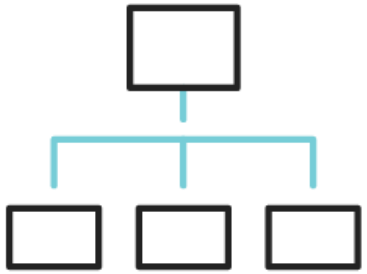
PoC

Production

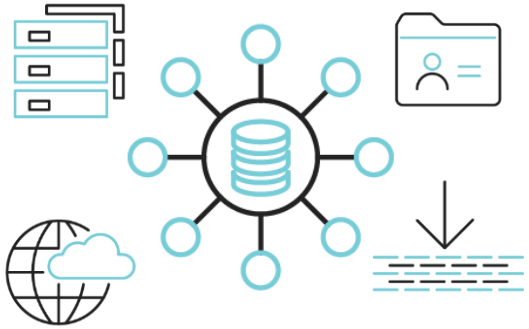
Development



Architecture



Customer's Data



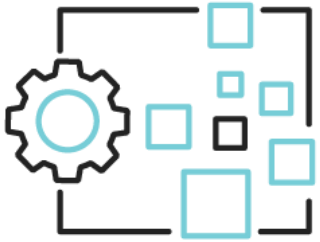
Feedback

# Data Science Project WorkFlow

Idea

DS team

Research



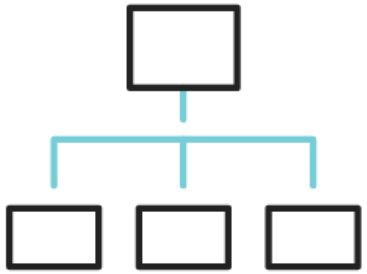
Concept

PoC

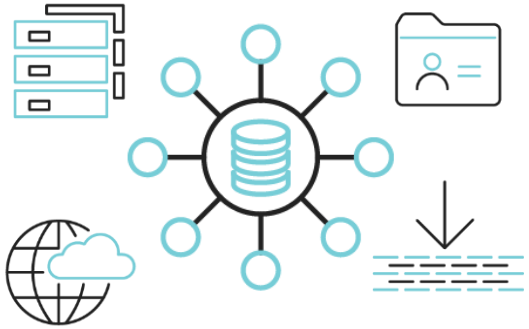
Production

Development


Architecture

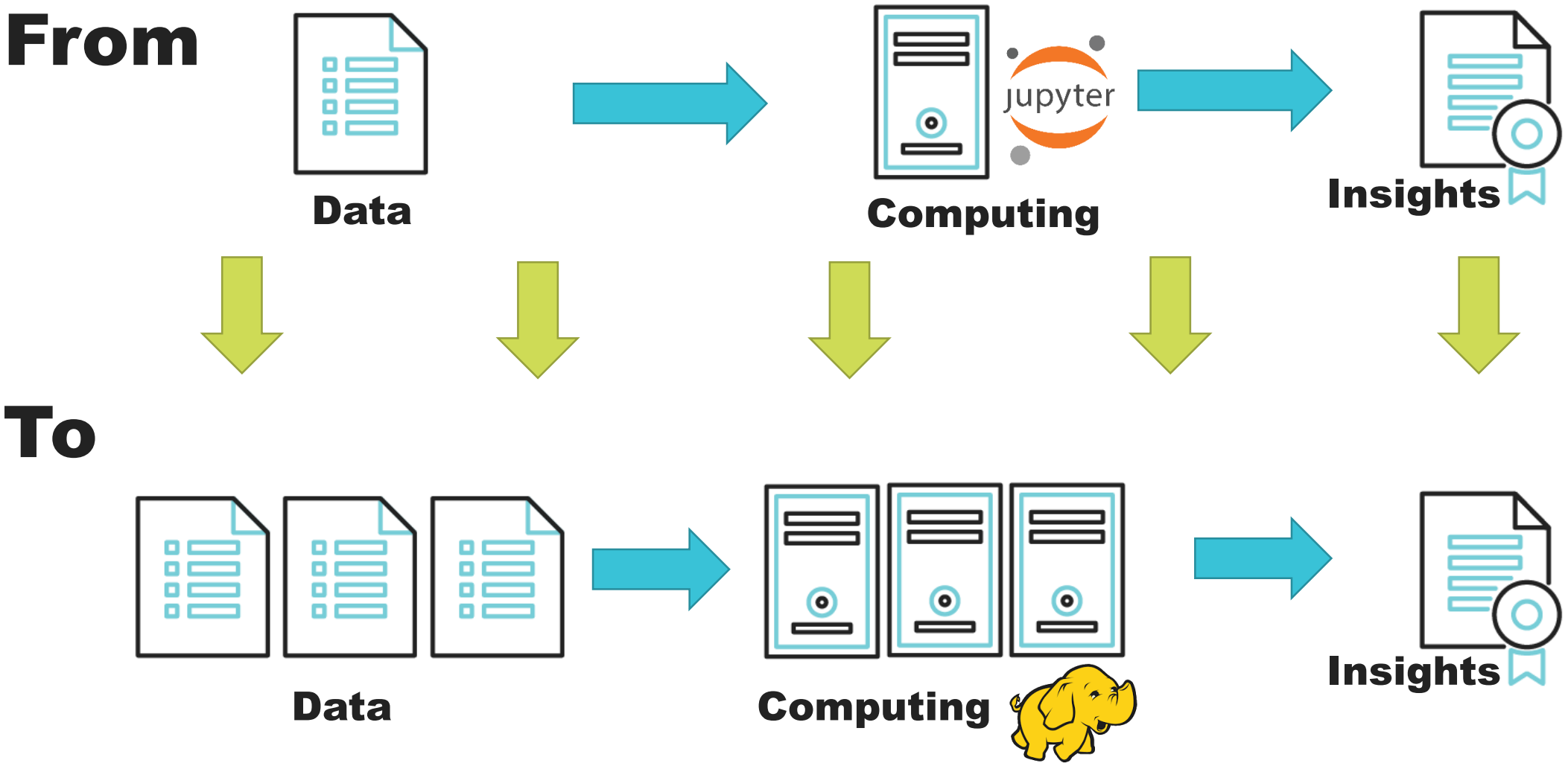


Customer's Data



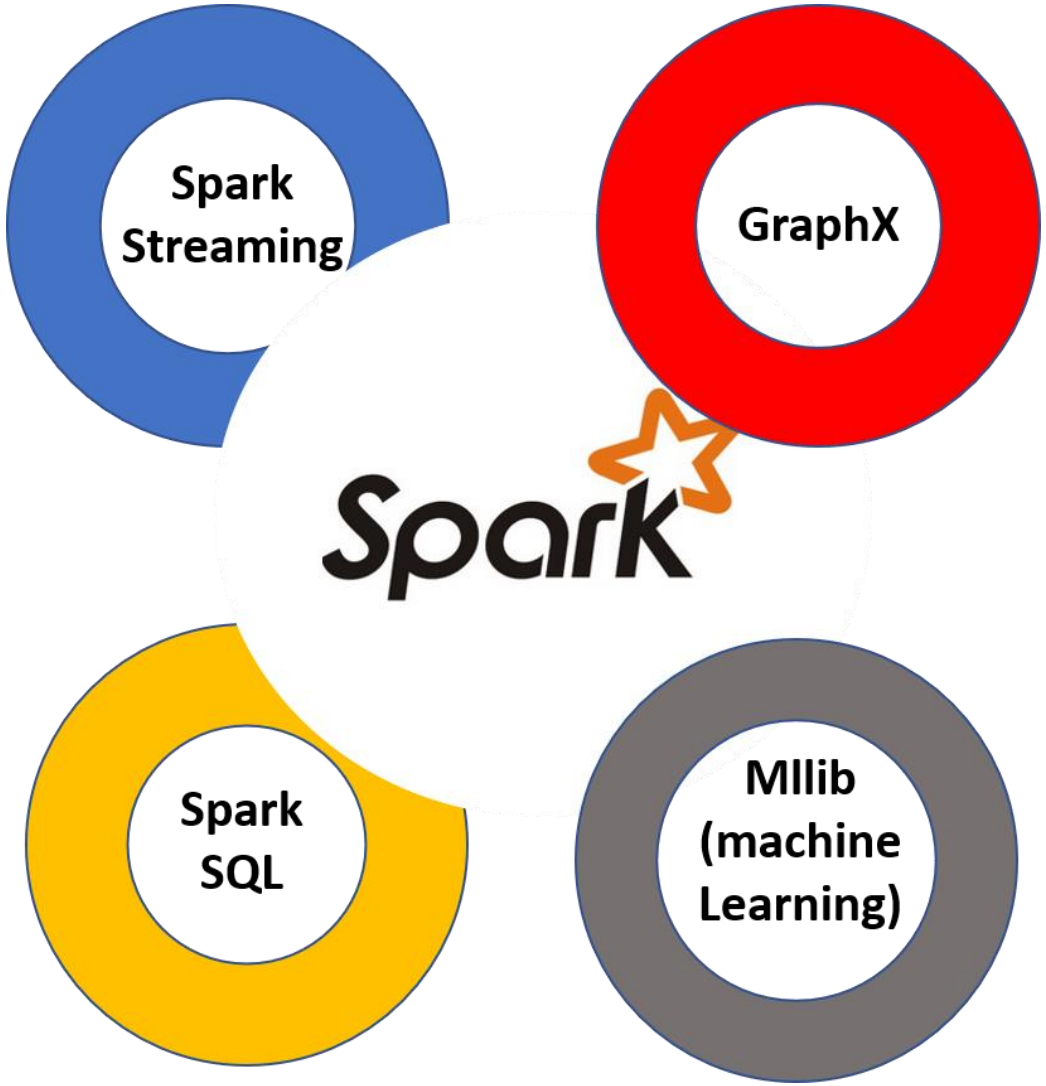
Feedback

# Distributed Environment Challenge



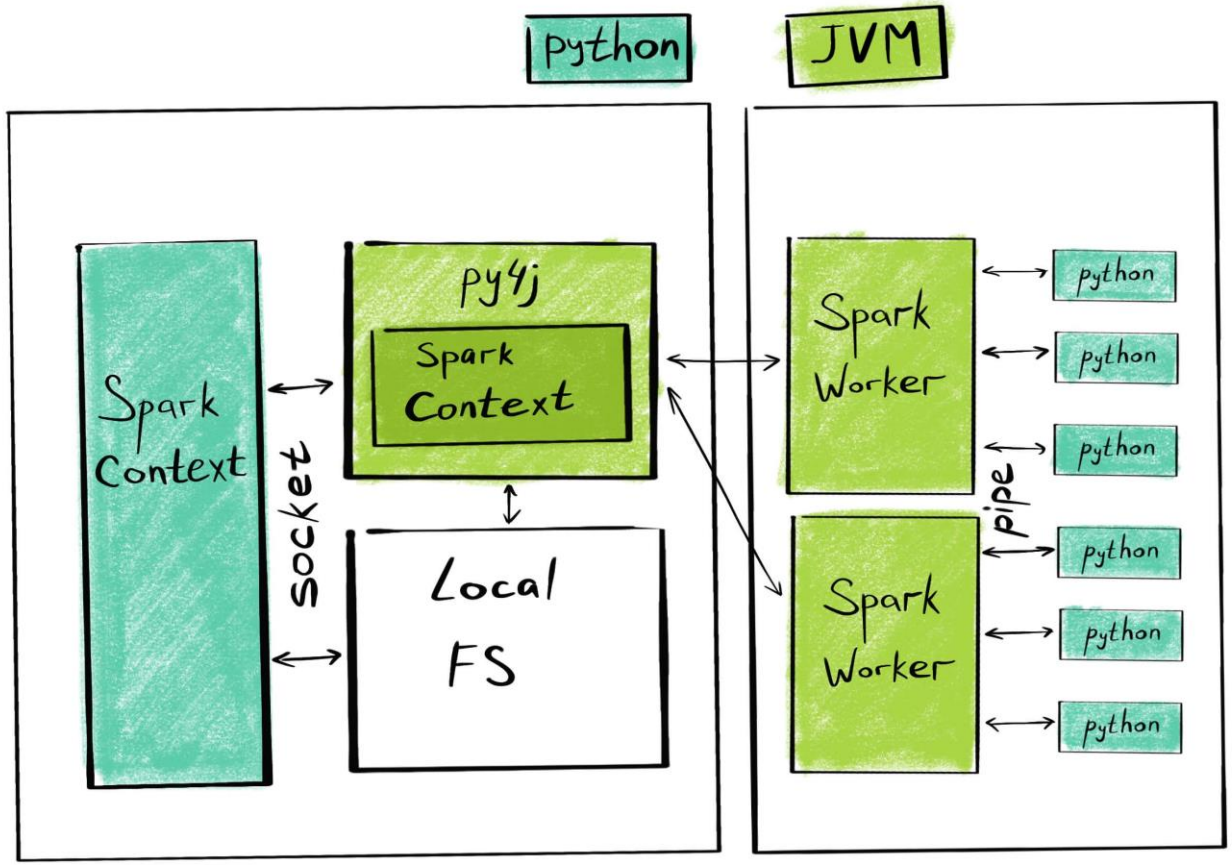
# What is Spark?

- Spark is a fast growing and general engine for **large-scale data processing**
- Runs **on Hadoop**, Mesos, standalone, or in the cloud
- Support for many programming languages **(Python)**
- **SQL**, streaming, and complex analytics
- Multiple options and libraries **(MLlib)**



# What is PySpark?

- Use **DataFrame** API
  - Efficient serialization/deserialization
  - Optimizations
  - Primary API for MLlib
- Do **NOT** Use **UDF**




@luminousmen.com

# Pipeline

Data


ETL

Azure

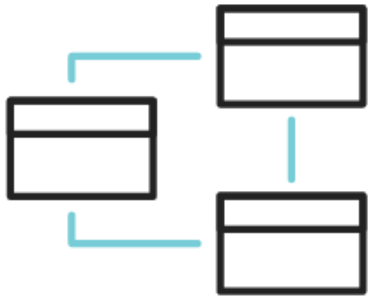
# Pipeline

**Data**

Azure

PySpark

**ETL**



**Models**

Keras

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

scikit learn

TensorFlow

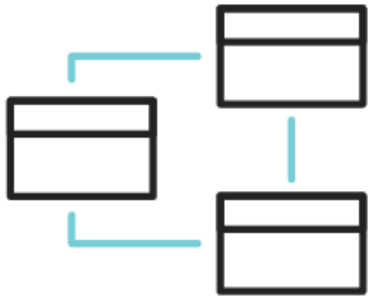
# Pipeline

**Data**

Azure

PySpark

**ETL**



**Models**

Keras

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

scikit learn

TensorFlow

PySpark

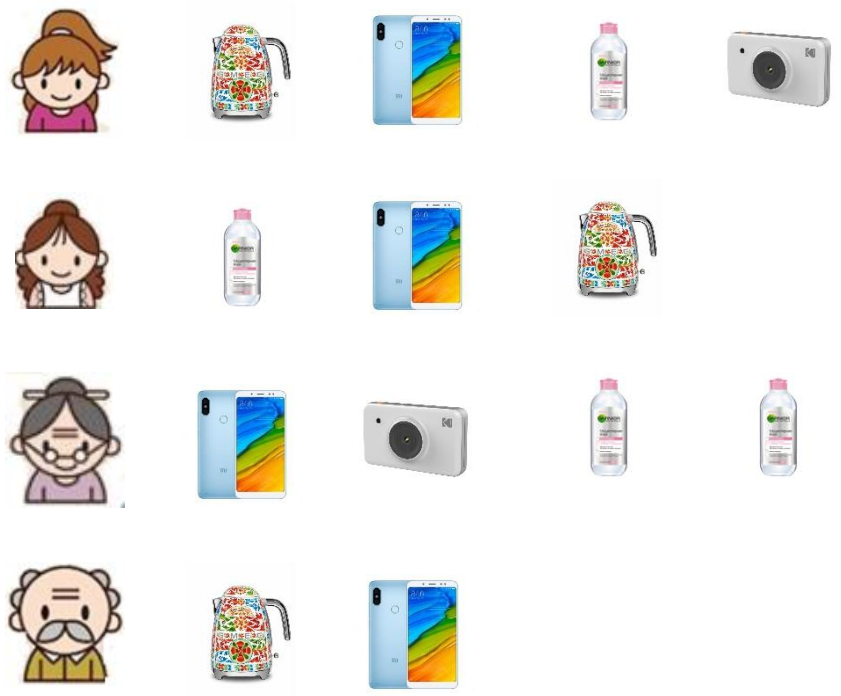
**Postprocessing**

**Insights**

Azure

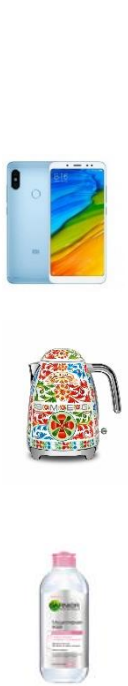
# Default Model

## Default model – based on recommendation of most popular products



**Users purchases**


**Count users that purchased set product**



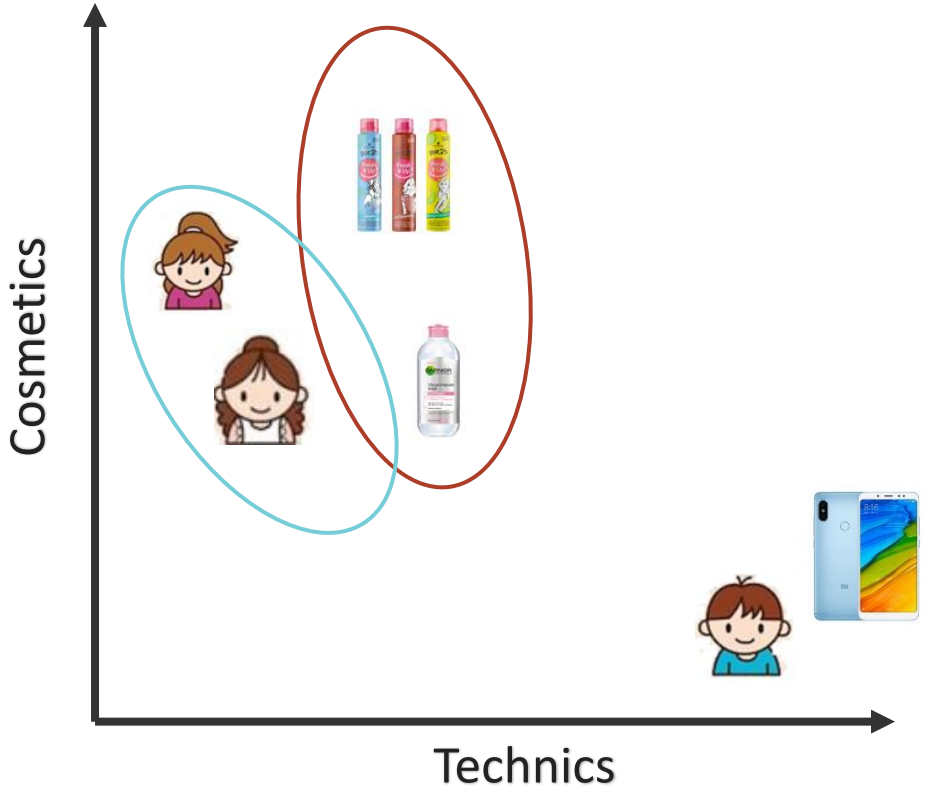
**Get top N=3 products**

# Collaborative Filtering. Matrix decomposition


**User-products matrix**

| | | | |
|---|---|---|---|
| | 0 | 0 | 2 |
| | 3 | 2 | 0 |
| | 2 | 1 | 0 |

=



# Collaborative Filtering. Product ranks calculation

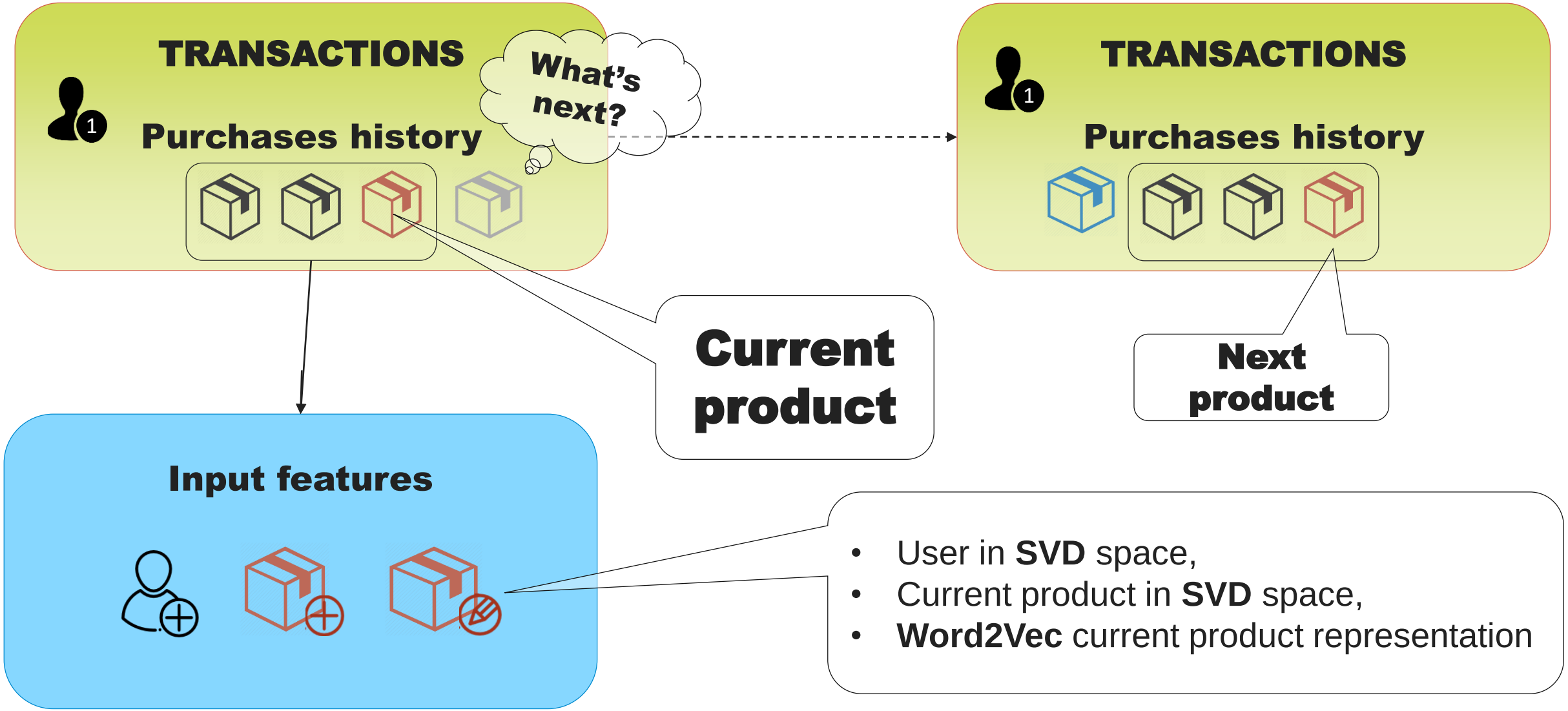
1.  2.  3.  for 

| | Cosmetics | Technics |
|---|---|---|
|  | 1.96 | 0.03 |
|  | 0.02 | 3.00 |
|  | 4.12 | 0.05 |

| |  |  |  |
|---|---|---|---|
| Cosmetics | 0.01 | 0.19 | 0.98 |
| Technics | 1.03 | 0.00 | 0.02 |

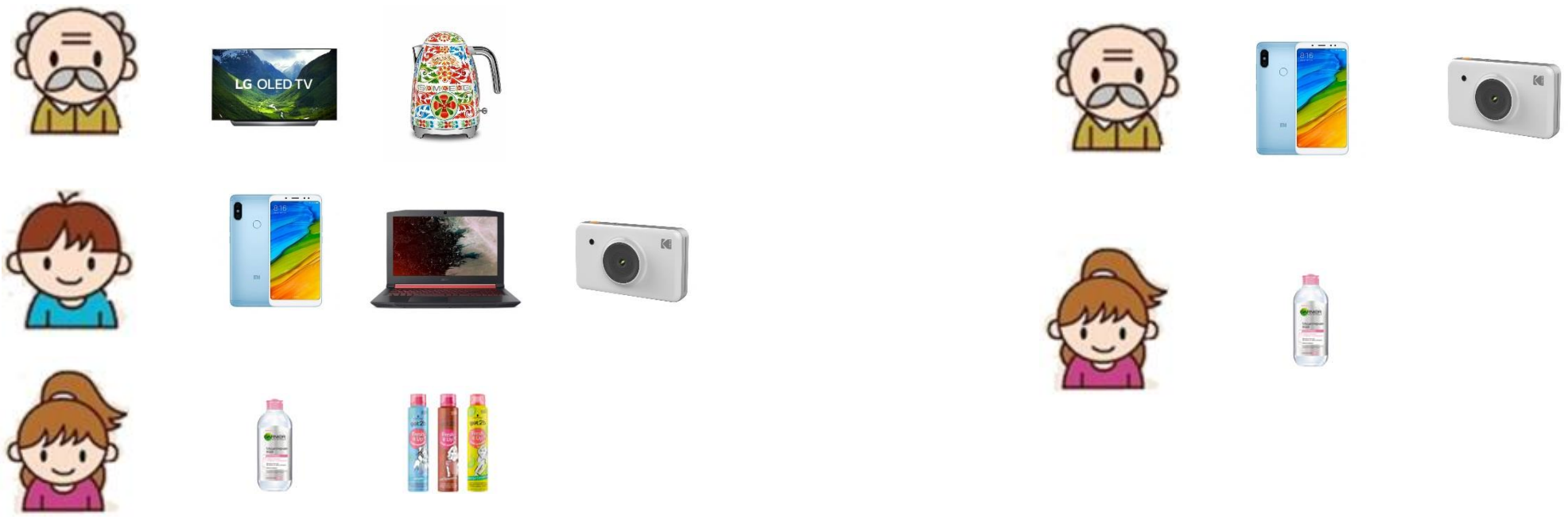
| |  |  |  |
|---|---|---|---|
|  | 0.05 | 0.37 | 1.92 |
|  | 3.09 | 0.00 | 0.08 |
|  | 0.09 | 0.78 | 4.04 |

# Neural Network Model



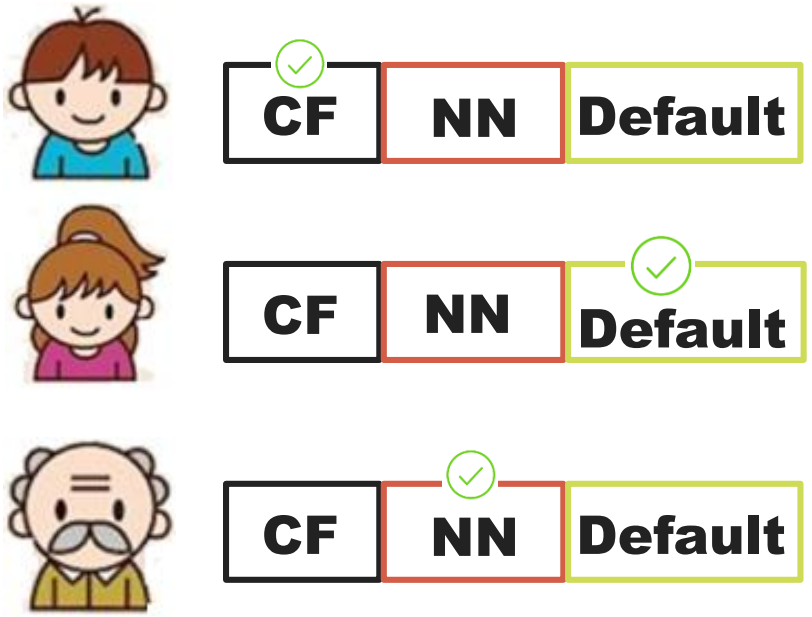
# King Of The Hill. Train Phase

| Train set | Test set |
| --- | --- |



# King Of The Hill. Train Phase

**Train set**



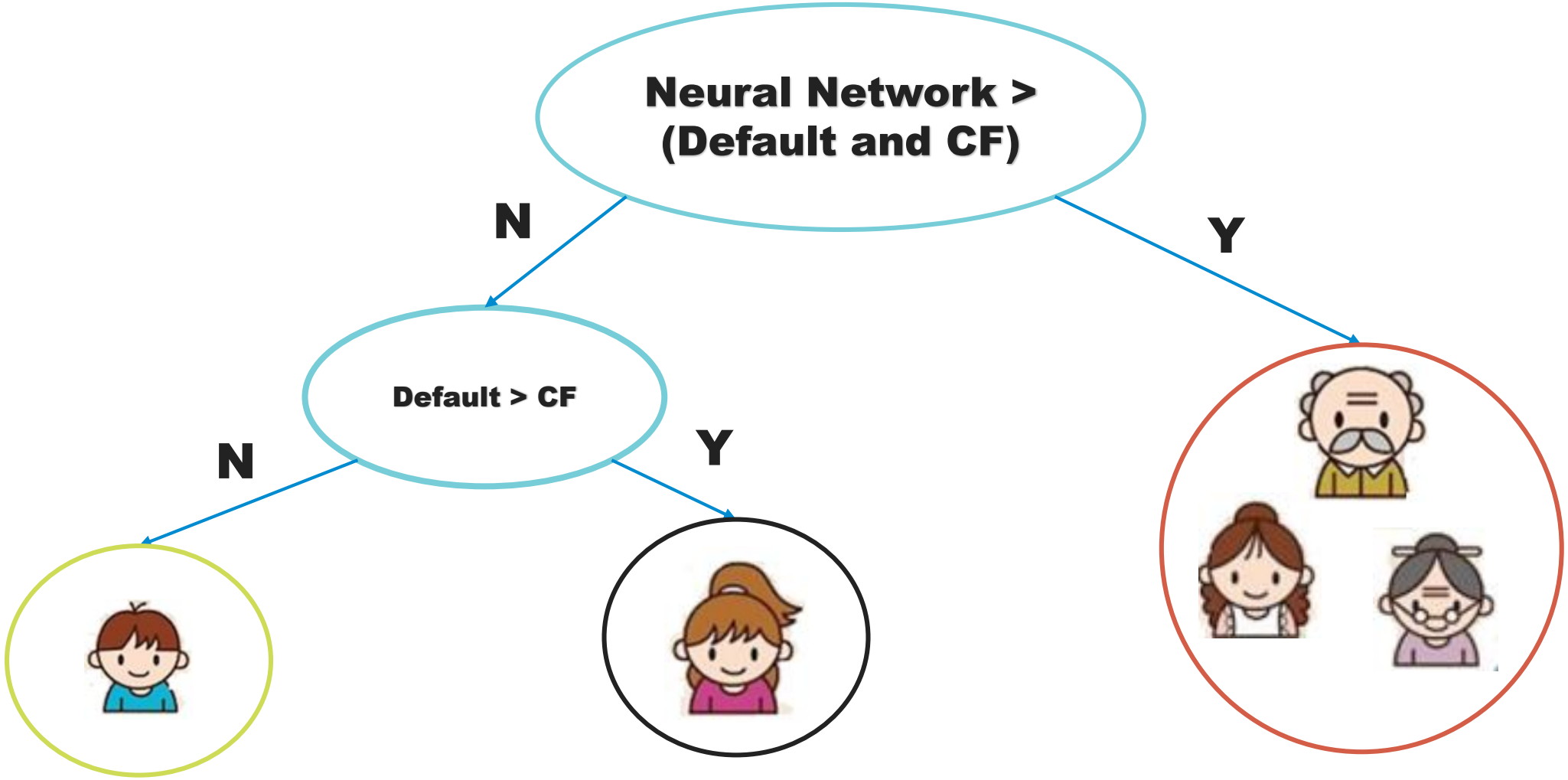

## Word2Vec

- Train representation for each product by word2vec

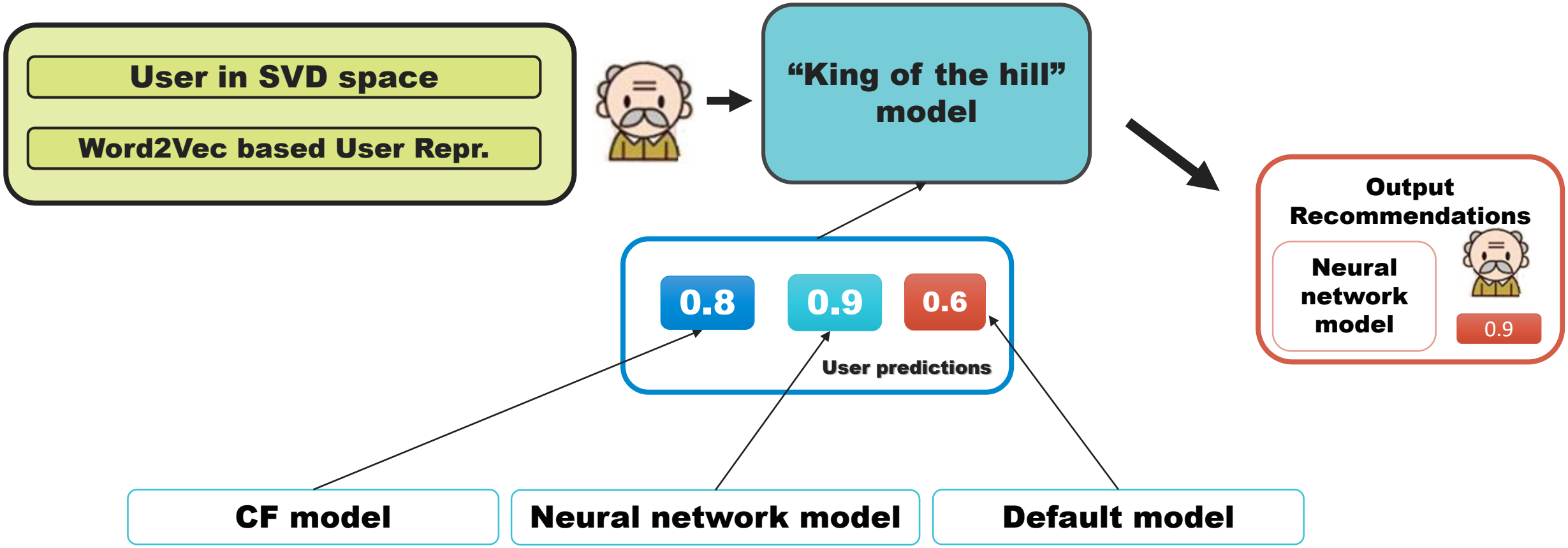
## AveragePrecision

- Selecting the best model for each user, based on scores for Test set

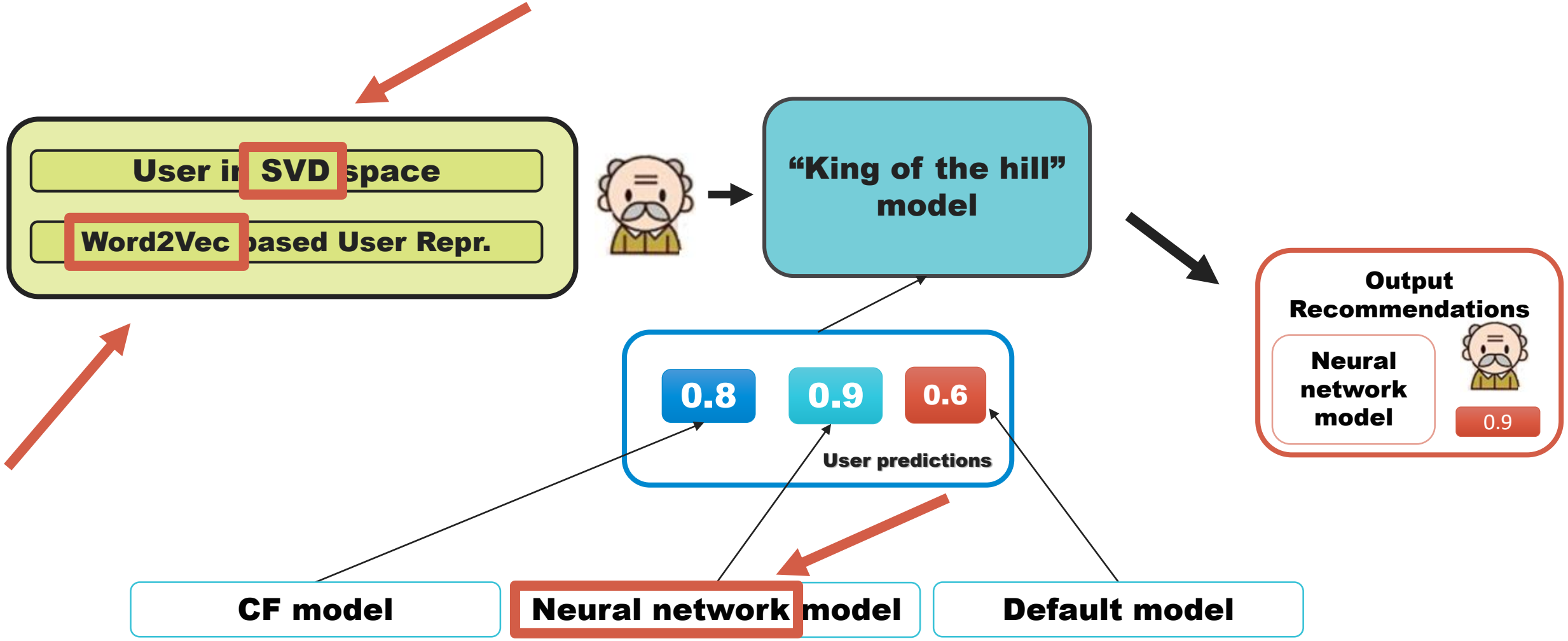
# King Of The Hill. Prediction Phase



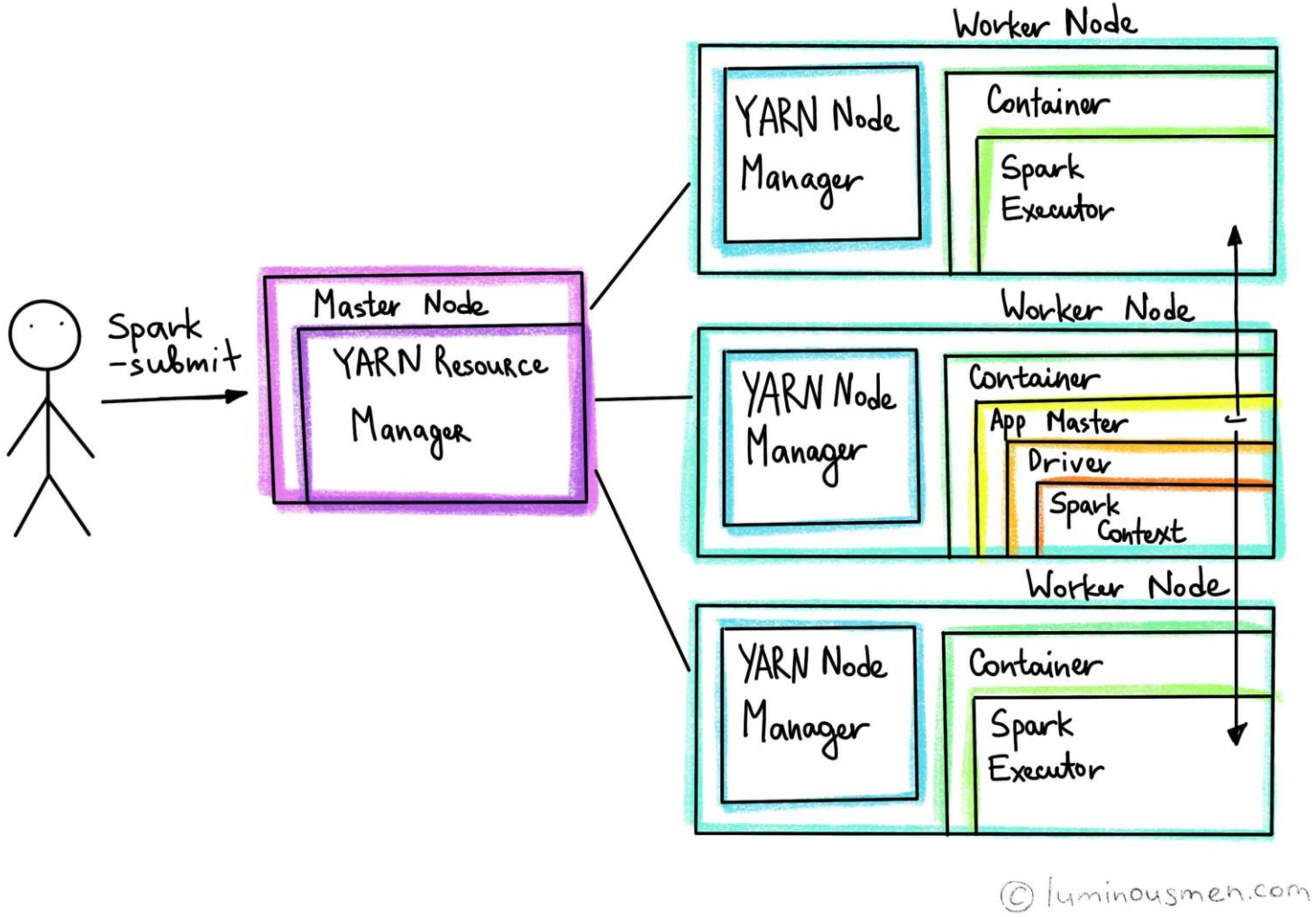
# King Of The Hill. Prediction Phase



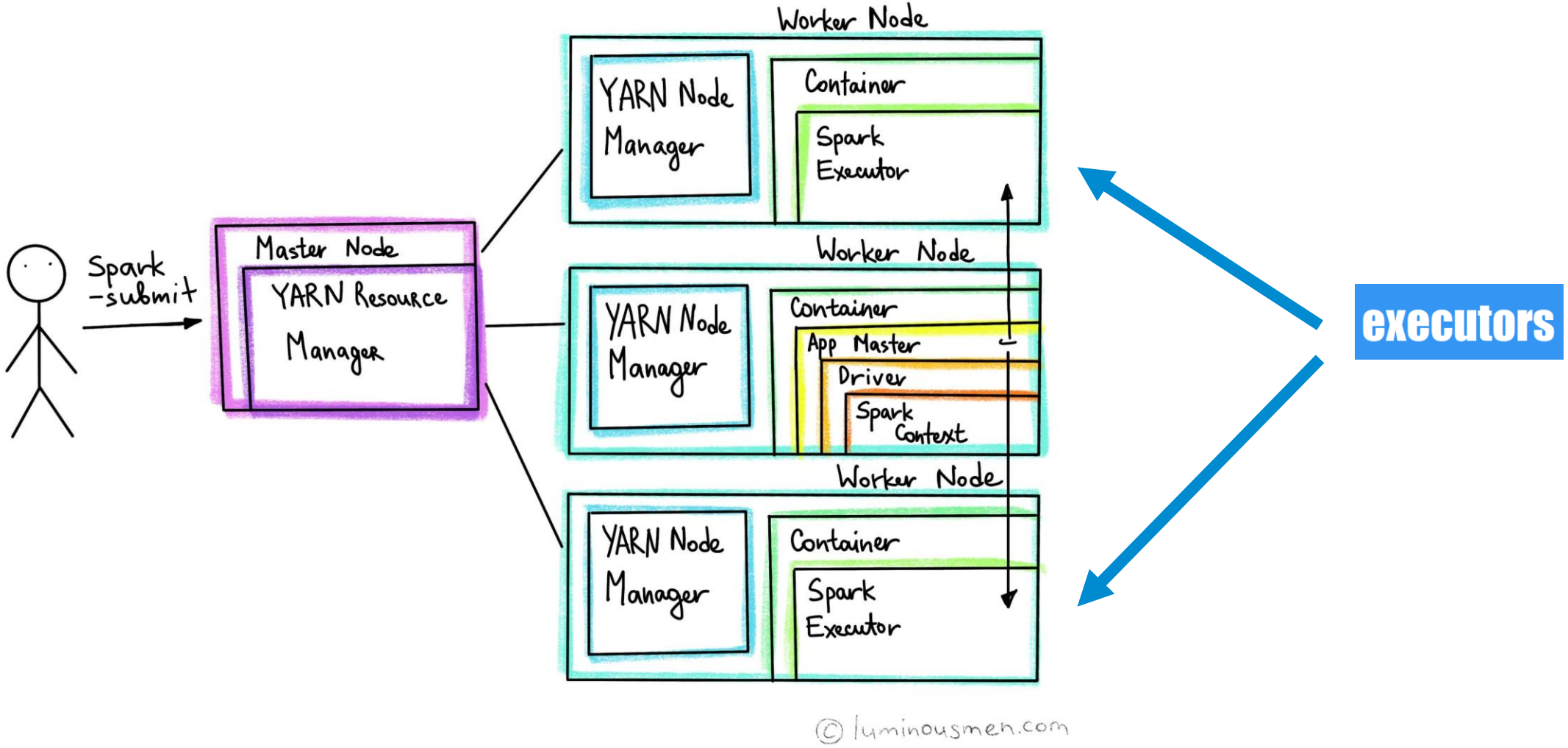
# King Of The Hill. Prediction Phase



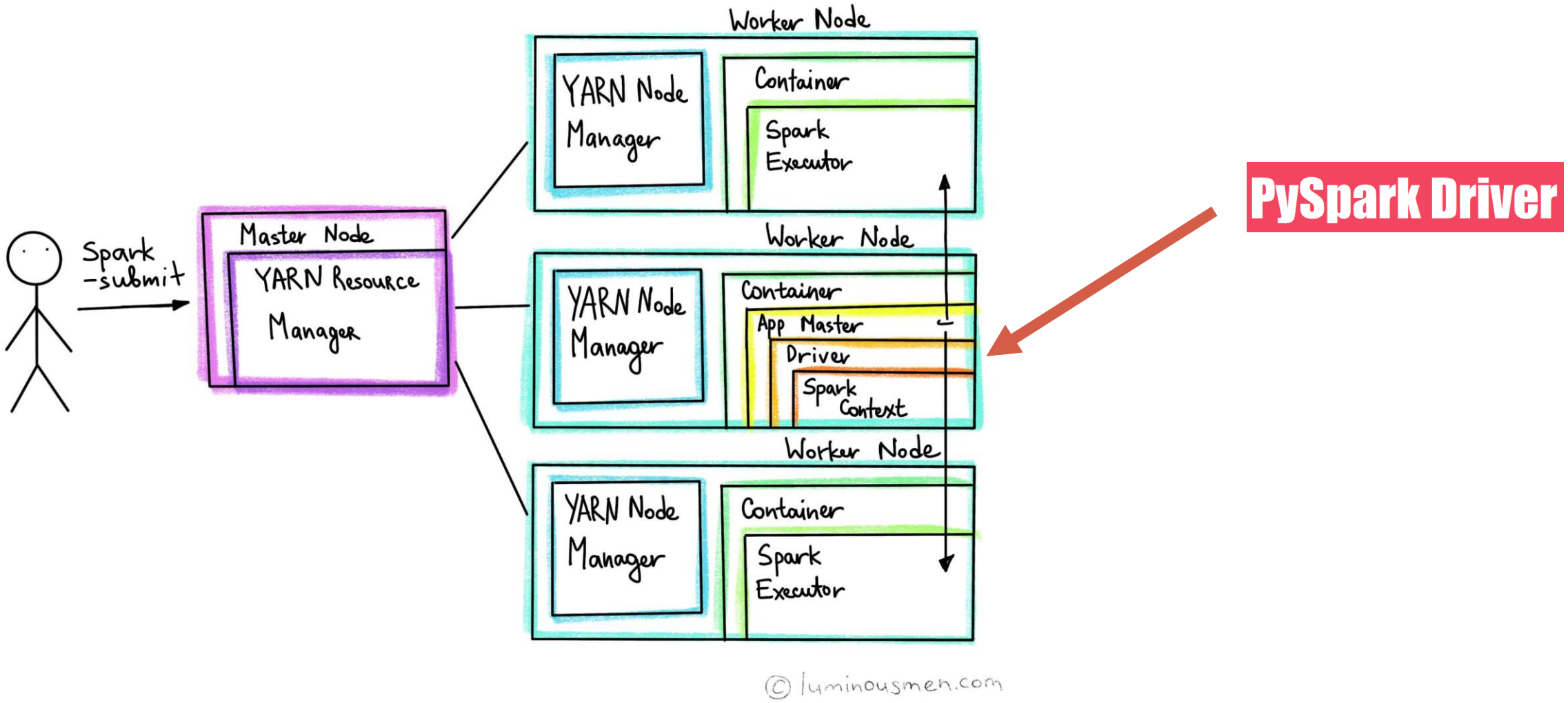
# Anatomy of Spark Application



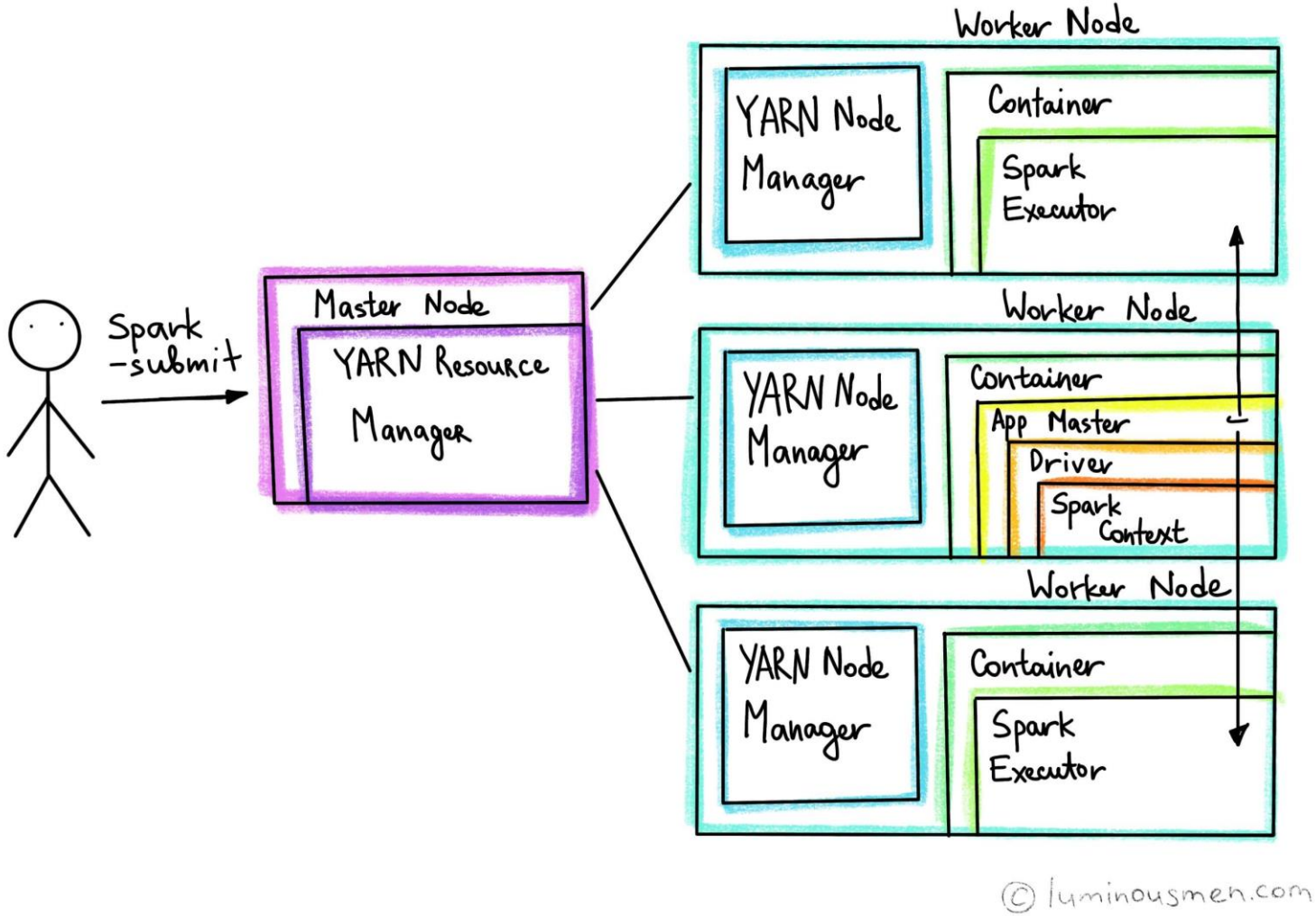
# Anatomy of Spark Application



# Anatomy of Spark Application



# Anatomy of Spark Application




# From Pandas to Spark. Models

1. W2V: gensim
2. SVD: scikit learn
3. Neural Network : K

PySpark

pandas

# From Pandas to Spark. Models

1. W2V: gensim
2. SVD: scikit learn
3. Neural Network : K

PySpark

pandas

# Word2Vec. Gensim VS Spark

- **size**
- **window**
- **min_count**
- **workers**
- **maxIter**
- **numPartitions**

```
model_w2v_prod = GensimWord2Vec(
    words_d, size=size, window=10, min_count=1, workers=14
)
return model_w2v_prod
```

```
word2vec = SparkWord2Vec(
    inputCol=target_field,
    vectorSize=size,
    minCount=1,
    windowSize=10,
    maxIter=num_partitions,  # maxIter <= numPartitions (ref to spark docs)
    numPartitions=num_partitions
)
return word2vec.fit(X.select(target_field))
```

# Word2Vec. Pitfalls

| | |
|---|---|
| Word2Vec | **setNumIterations**(int numIterations)<br>Sets number of iterations (default: 1), which should be smaller than or equal to number of partitions. |
| Word2Vec | **setNumPartitions**(int numPartitions)<br>Sets number of partitions (default: 1). |

https://spark.apache.org/docs/2.4.3/api/java/org/apache/spark/mllib/feature/Word2Vec.html

# Word2Vec. Pitfalls

| | |
|---|---|
| Word2Vec | **setNumIterations**(int numIterations)<br>Sets number of iterations (default: 1), which should be smaller than or equal to number of partitions. |
| Word2Vec | **setNumPartitions**(int numPartitions)<br>Sets number of partitions (default: 1). |

```
sg : int {1, 0}
    Defines the training algorithm. If 1, skip-gram is employed; otherwise, CBOW is used.
```

https://spark.apache.org/docs/2.4.3/api/java/org/apache/spark/mllib/feature/Word2Vec.html

# Word2Vec. Pitfalls

| | |
|---|---|
| Word2Vec | **setNumIterations**(int numIterations)<br>Sets number of iterations (default: 1), which should be smaller than or equal to number of partitions. |
| Word2Vec | **setNumPartitions**(int numPartitions)<br>Sets number of partitions (default: 1). |

```
sg : int {1, 0}
    Defines the training algorithm. If 1, skip-gram is employed; otherwise, CBOW is used.
```

```
from pyspark.ml.feature import Word2Vec as mlWord2Vec, Word2VecModel

from pyspark.mllib.feature import Word2Vec as mllibWord2Vec, Word2VecModel
```

https://spark.apache.org/docs/2.4.3/api/java/org/apache/spark/mllib/feature/Word2Vec.html

# Word2Vec. Pitfalls

| | |
|---|---|
| Word2Vec | **setNumIterations**(int numIterations)<br>Sets number of iterations (default: 1), which should be smaller than or equal to number of partitions. |
| Word2Vec | **setNumPartitions**(int numPartitions)<br>Sets number of partitions (default: 1). |

```
sg : int {1, 0}
    Defines the training algorithm. If 1, skip-gram is employed; otherwise, CBOW is used.
```

```
from pyspark.ml.feature import Word2Vec as mlWord2Vec, Word2VecModel

from pyspark.mllib.feature import Word2Vec as mllibWord2Vec, Word2VecModel
```

https://spark.apache.org/docs/2.4.3/api/java/org/apache/spark/mllib/feature/Word2Vec.html

# Word2Vec. Performance

| | pysprk.ml.feature.Word2vec | gensim.models. Word2vec |
|---|---|---|
| m@pk | 0.083 | 0.1 |

| num partitions | 4 | | 5 | |
|---|---|---|---|---|
| | m@pk | time | m@pk | time |
| avg | 0.075 | 54.578 min | 0.062 | 53.465 min |

# From Pandas to Spark. Models

1. W2V: gensim

2. SVD: scikit learn

3. Neural Network : K

PySpark

pandas

# TruncatedSVD. Sklearn

```python
def get_model(self, config):
    """Get module sub model"""
    return TruncatedSVD(**config)


def get_default_config(self):
    """Get default model configuration"""
    return {
        'random_state': 12345,
        'algorithm': 'randomized', ## R-SVD
        'n_iter': 10,
        'n_components': 300
    }
```

```python
self.row_index2pos, row_pos2index, row_pos = convert_to_idx_encodings(X[self.user_field])
self.col_index2pos, col_pos2index, col_pos = convert_to_idx_encodings(X[self.target_field])

n_rows = row_pos2index.shape[0]
n_cols = col_pos2index.shape[0]

logger.info("Creating sparse matrix...")
self.matrix = csr_matrix((X[value_field], (row_pos, col_pos)), shape=(n_rows, n_cols))
logger.info("Sparse matrix size: {} ".format(self.matrix.shape))

self.user_index = row_pos2index
self.prom_index = col_pos2index

logger.info("Fitting model...")
self.model.fit(self.matrix)
```

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

# TruncatedSVD. Sklearn

```python
def get_model(self, config):
    """Get module sub model"""
    return TruncatedSVD(**config)


def get_default_config(self):
    """Get default model configuration"""
    return {
        'random_state': 12345,
        'algorithm': 'randomized', ## R-SVD
        'n_iter': 10,
        'n_components': 300
    }
```

```python
self.row_index2pos, row_pos2index, row_pos = convert_to_idx_encodings(X[self.user_field])
self.col_index2pos, col_pos2index, col_pos = convert_to_idx_encodings(X[self.target_field])

n_rows = row_pos2index.shape[0]
n_cols = col_pos2index.shape[0]

logger.info("Creating sparse matrix...")
self.matrix = csr_matrix((X[value_field], (row_pos, col_pos)), shape=(n_rows, n_cols))
logger.info("Sparse matrix size: {} ".format(self.matrix.shape))

self.user_index = row_pos2index
self.prom_index = col_pos2index

logger.info("Fitting model...")
self.model.fit(self.matrix)
```

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

# ComputeSvd. Spark

```python
def fit(self):
    """
    Calls computeSVD method for
    IndexedRowMatrix if matrix is not None
    """
    if self._mat:
        logger.info("computeSVD with {} components"
                    .format(self.n_components))
        self.svd = self._mat.computeSVD(self.n_components,
                                        computeU=True)
        logger.info("SVD computed")
    else:
        logger.warning("Matrix is not provided")
    return self.svd
```

```python
sdf_with_indexes = X \
    .join(row_index_mapping, on=user_field) \
    .join(col_index_mapping, on=target_field)
sdf_with_indexes = sdf_with_indexes.select(row_index_field,
                                           col_index_field,
                                           value_field)

num_products = col_index_mapping.count()
rows = sdf_with_indexes.rdd \
    .map(lambda r: (r[0], [(r[1], r[2])])) \
    .reduceByKey(lambda x, y: x + y)

with_indexed_rows = rows.map(
    lambda r: IndexedRow(r[0], Vectors.sparse(num_products,
                         sorted(r[1], key=itemgetter(0)))))
return IndexedRowMatrix(with_indexed_rows)
```

https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.linalg.distributed.IndexedRowMatrix.computeSVD

# ComputeSvd. Spark

```python
def fit(self):
    """
    Calls computeSVD method for
    IndexedRowMatrix if matrix is not None
    """
    if self._mat:
        logger.info("computeSVD with {} components"
                    .format(self.n_components))
        self.svd = self._mat.computeSVD(self.n_components,
                                        computeU=True)
        logger.info("SVD computed")
    else:
        logger.warning("Matrix is not provided")
    return self.svd
```

```python
sdf_with_indexes = X \
    .join(row_index_mapping, on=user_field) \
    .join(col_index_mapping, on=target_field)
sdf_with_indexes = sdf_with_indexes.select(row_index_field,
                                           col_index_field,
                                           value_field)

num_products = col_index_mapping.count()
rows = sdf_with_indexes.rdd \
    .map(lambda r: (r[0], [(r[1], r[2])])) \
    .reduceByKey(lambda x, y: x + y)

with_indexed_rows = rows.map(
    lambda r: IndexedRow(r[0], Vectors.sparse(num_products,
                         sorted(r[1], key=itemgetter(0)))))
return IndexedRowMatrix(with_indexed_rows)
```

https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.linalg.distributed.IndexedRowMatrix.computeSVD

# SVD. Performance

| | Spark SVD | scikit-learn SVD | | | |
|---|---|---|---|---|---|
| Data | Full | Full | Sample | | |
| | | | 2% | 5% | 10% |
| Sparse matrix size: (74279, 17757) | 0.073 | **0.084** | 0.001 | 0.003 | 0.005 |
| Sparse matrix size: (1818114, 56489) | 0.102 | **0.13** | 0.002 | 0.006 | 0.011 |

https://databricks.com/blog/2014/07/21/distributing-the-singular-value-decomposition-with-spark.html

# From Pandas to Spark. Models

1. W2V: gensim
2. SVD: scikit learn
3. Neural Network : K

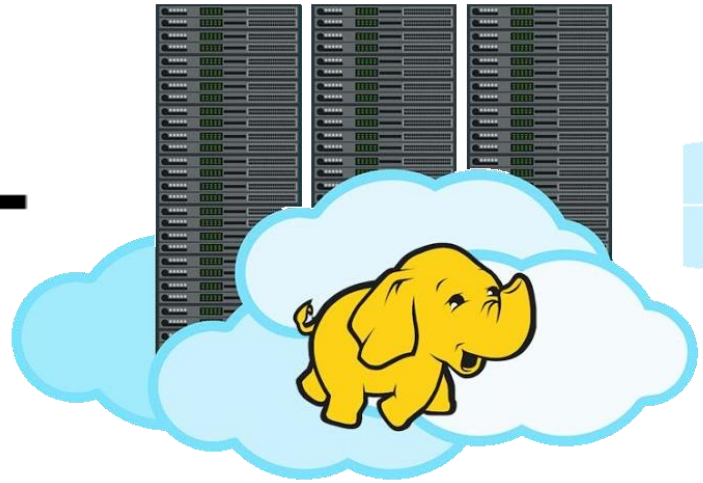
PySpark

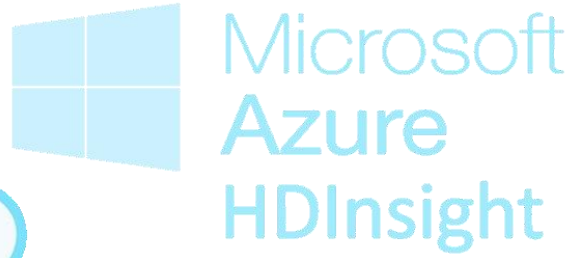
pandas

# Neural Networks

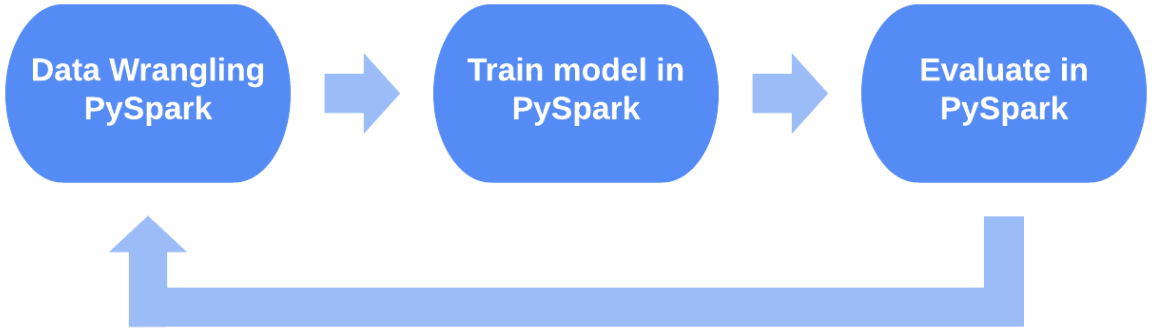

+

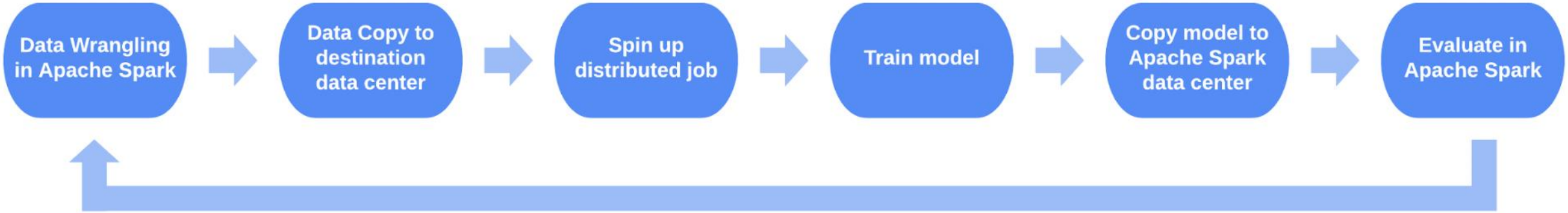

+





# HOROVOD

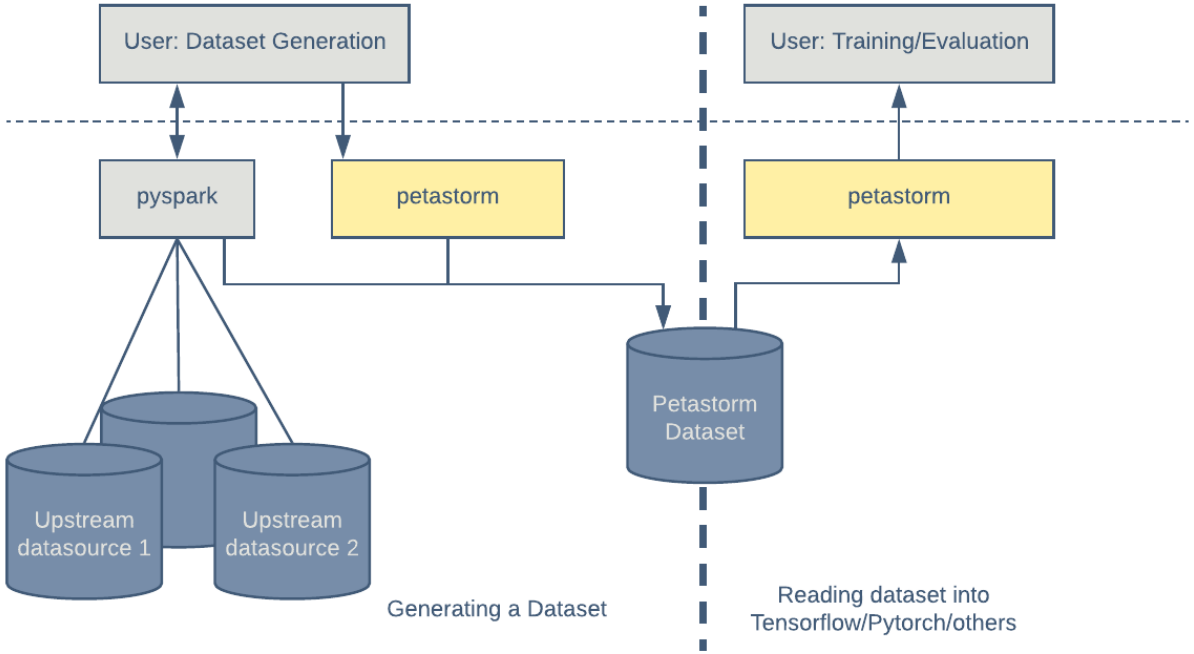




https://eng.uber.com/horovod-pyspark-apache-mxnet-support

# Petastorm






Only HDFS or S3 distributed filesystems can be used
**(we're using Azure and “wasbs://” protocol)**

https://eng.uber.com/petastorm

# Petastorm

Apache Arrow

Issues

Reports

Components

Apache Arrow / ARROW-2034

## [C++] Filesystem implementation for Azure Blob Store

Export

**Details**

| | | | |
|---|---|---|---|
| Type: | New Feature | Status: | OPEN |
| Priority: | Major | Resolution: | Unresolved |
| Affects Version/s: | None | Fix Version/s: | None |
| Component/s: | C++ | | |
| Labels: | filesystem | | |

**People**

| | |
|---|---|
| Assignee: | Uwe L. Korn |
| Reporter: | Wes McKinney |
| Votes: | 0 Vote for this issue |
| Watchers: | 2 Start watching this issue |

**Dates**

| | |
|---|---|
| Created: | 25/Jan/18 14:55 |
| Updated: | 29/Mar/19 19:48 |

**Activity**

All | Comments | Work Log | History | Activity | Transitions

There are no comments yet on this issue.

https://issues.apache.org/jira/browse/ARROW-2034

https://arrow.apache.org/docs/python/parquet.html#reading-a-parquet-file-from-azure-blob-storage

# From Pandas to Spark. Models

1 W2V: gensim

2 SVD: scikit learn

3 Neural Network : K

# From Pandas to Spark. Models

1 W2V: gensim → Spark MLlib

2 SVD: scikit learn → Spark MLlib

3 Neural Network : K → HOROVOD

# From Pandas to Spark. Models

1 W2V: gensim → Spark MLlib ✓

2 SVD: scikit learn → Spark MLlib ✓

3 Neural Network : K → HOROVOD !

# Questions Time

Andrei_Gavrilov@epam.com

@tbont