ORACLE

Oracle Machine Learning：企業向けのRおよびPythonのスケーリング

Mark Hornick
Marcos Arancibia

Oracle Machine Learning Product Management

免責条項

下記事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。マテリアルやコード、機能の提供をコミットメント（確約）するものではなく、購買を決定する際の判断材料になさらないでください。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

データ・サイエンティストやデータ・アナリストがRおよびPythonを使用する理由

強力

拡張可能

グラフィカル

広範な統計情報

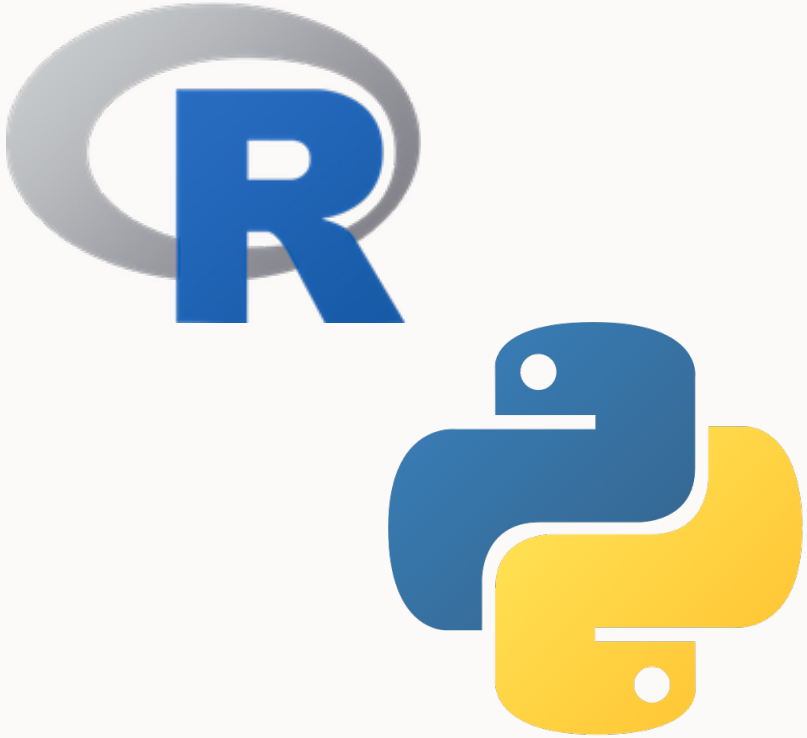
インストールのしやすさ、使いやすさ

充実したエコシステム

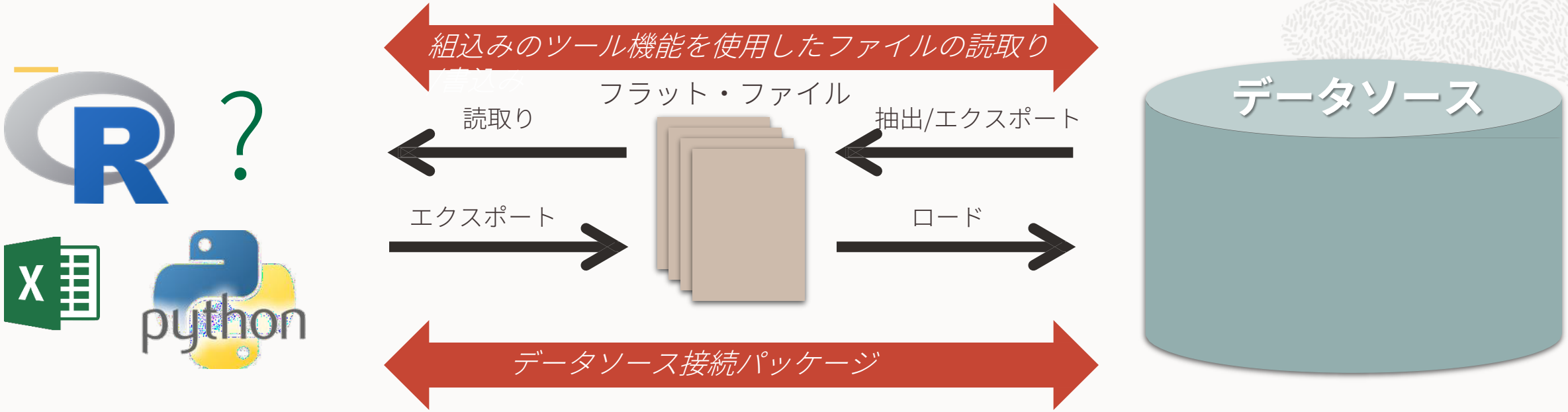
- 数千のオープン・ソース・パッケージ
- 世界中に数百万人のユーザー

データ・サイエンティストが頻繁に利用

無料



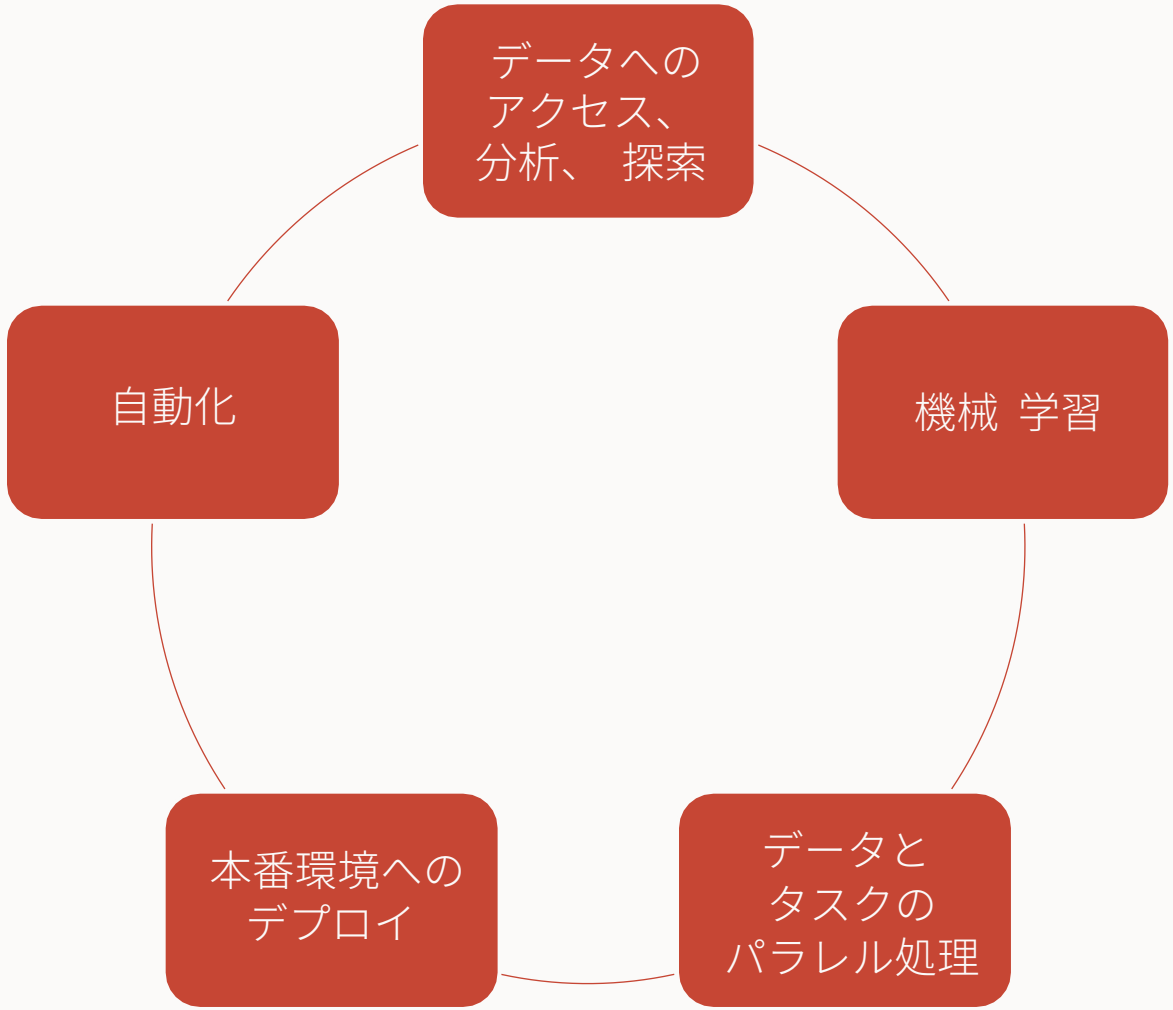
従来の分析とデータソースの相互作用



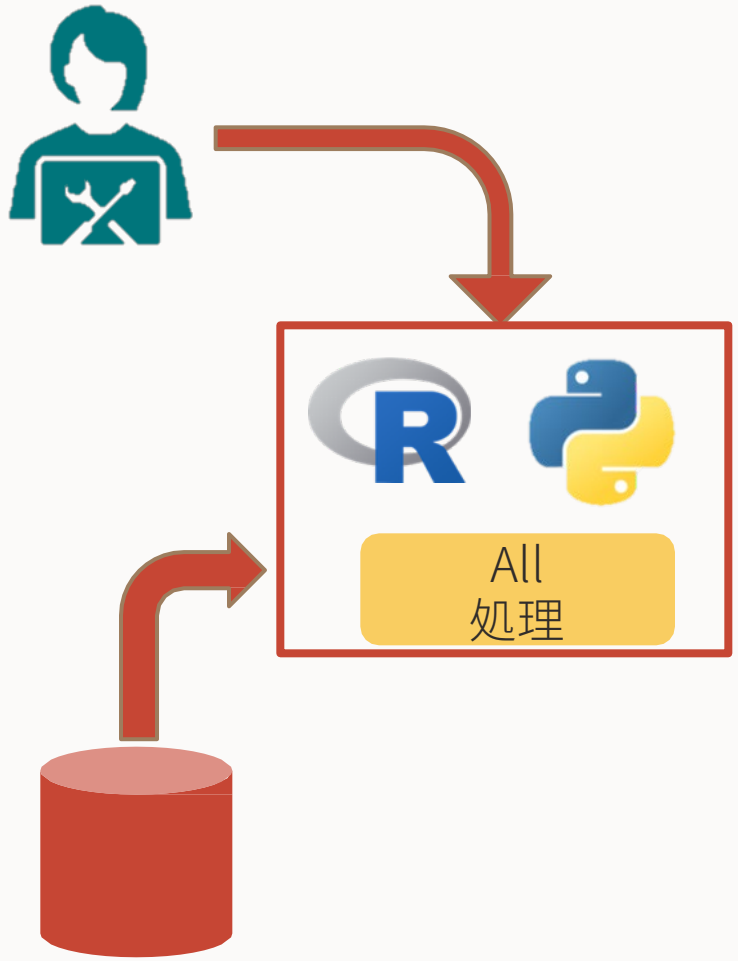
デプロイ
非定型
cronジョブ

アクセスの待機時間
パラダイム・シフト：R/Python→*データ・アクセス言語*→R/Python
メモリの制約 – データ・サイズ、インメモリ処理
シングルスレッド
バックアップ、リカバリ、セキュリティの問題
本番環境への非定型のデプロイ

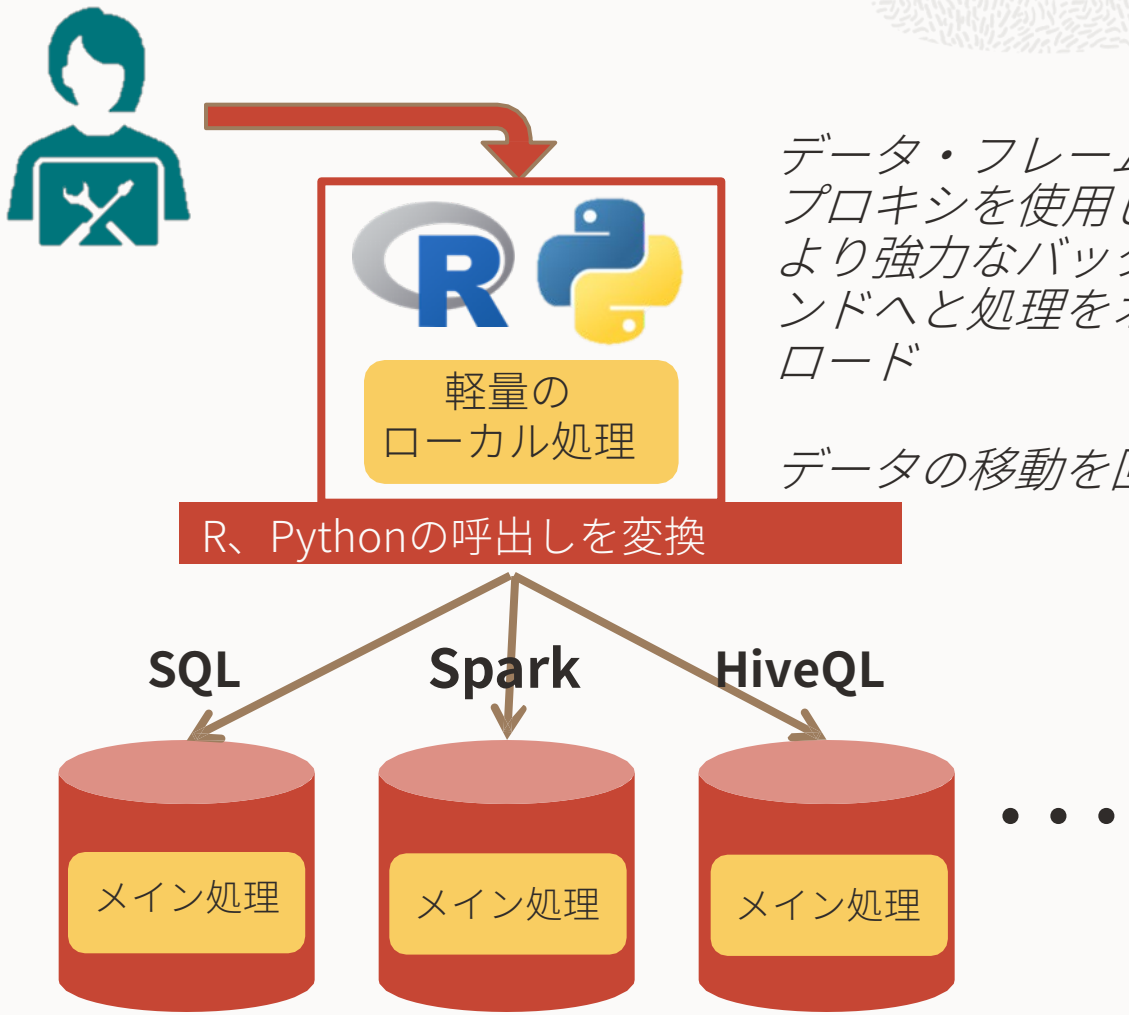
企業のスケーラビリティに影響する要素 - R、Pythonの場合



データへのアクセス、分析、探索



データソース



データ・フレーム・プロキシを使用してより強力なバックエンドへと処理をオフロード

データの移動を回避

データ + HPC環境

データへのアクセス、分析、探索

- 言語の機能やインタフェースをそのまま活用
- プロキシ・オブジェクト経由でデータを参照することでデータ移動を回避
- オープン・ソースの呼出しをデータ処理エンジンの言語に変換する関数をオーバーロード
- データ処理エンジンで実行

すべてのデータをより迅速に分析

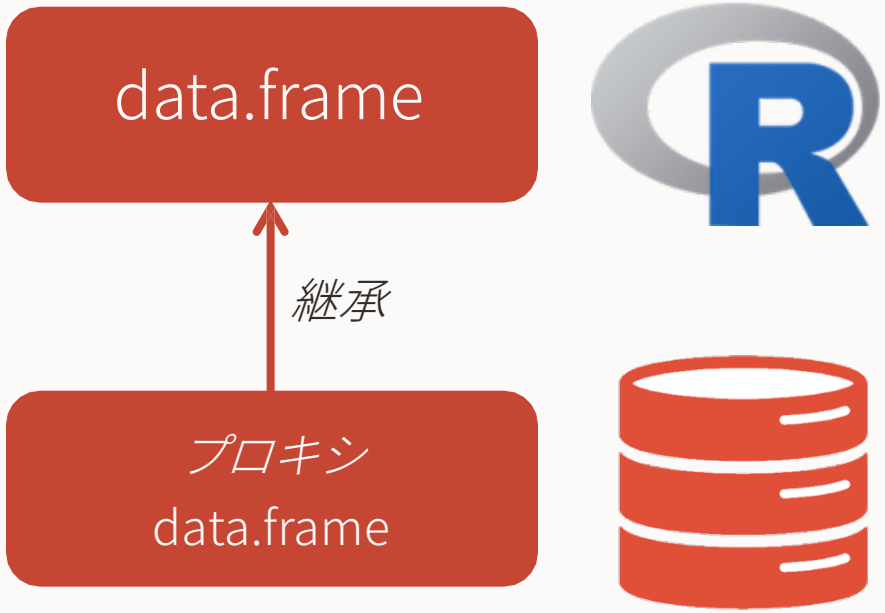
データ・アクセスの待機時間なし

データ処理エンジンのパフォーマンス最適化機能を活用

プロキシ・オブジェクト

OML4Rインタフェースを使用した例

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa




```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> str(IRIS)
'data.frame':   150 obs. of  5 variables:
Formal class 'ore.frame' [package "OREbase"] with 12 slots
  ..@ .Data    : list()
  ..@ dataQry  : Named chr "( select /*+ no_merge(t) */  \"Sepal.Length\" VAL001,\"Sepal.Wid
th\" VAL002,\"Petal.Length\" VAL003,\"Petal.Width\" VAL004,\"Sp"| __truncated__
  .. ..- attr(*, "names")= chr "2539_1"
  ..@ dataObj  : chr "2539_1"
  ..@ desc     :'data.frame':   5 obs. of  2 variables:
  .. ..$ name  : chr  "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" ...
  .. ..$ Sclass: chr  "numeric" "numeric" "numeric" "numeric" ...
  ..@ sqlName  : chr
  ..@ sqlValue : chr  "\"Sepal.Length\"" "\"Sepal.Width\"" "\"Petal.Length\"" "\"Petal.Width
```

```
"( select /*+ no_merge(t) */  \"Sepal.Length\" VAL001,\"Sepal.Width\" VAL
002,\"Petal.Length\" VAL003,\"Petal.Width\" VAL004,\"Species\" VAL005 fro
m \"RQUSER\".\"IRIS\" t  )"
```

オープン・ソース言語によるOracle Databaseへのアクセス

OML4R

```
library(ORE)



ore.connect("oml_user",...)
```

```
ore.sync()
ore.attach()
ore.ls()
head(ONTIME_S)
TMP_ONTIME <- ore.get("ONTIME_S")
```

```
dim(ONTIME_S)
summary(ONTIME_S)
cor(ONTIME_S[,c('ARRDELAY','DEPDELAY')],
    use="complete.obs")
```

OML4Py*

```
import oml
import pandas as pd
import matplotlib.pyplot as plt
oml.connect("oml_user",...)
```

```
t = oml.sync(table="*",regex_match=True)
t.keys()
oml.dir()
ONTIME_S = t.get("ONTIME_S","none")
ONTIME_S.head()
```

```
ONTIME_S.shape()
ONTIME_S.describe()
ONTIME_S[['ARRDELAY','DEPDELAY']].corr()
```


* Pythonについては近日追加予定

オープン・ソース言語によるOracle Databaseへのアクセス

OML4R

```
DF <-
    ONTIME_S[ONTIME_S$DEST=="SFO",1:21])
```

```
hist(DF$ARRDELAY,breaks=100,color="green",
        main= 'Histogram of Arrival Delay',
        xlab = 'Arrival Delay (minutes)',
        ylab = '# of flights')
```

```
boxplot(SEPAL_WIDTH ~ Species, data=IRIS,
        notch=TRUE, ylab='cm',
        main= 'Distribution of IRIS Attributes')
```

OML4Py*

```
DF =
    ONTIME_S[ONTIME_S["DEST"]=="SFO",1:21]
```

```
_ = oml.graphics.hist(DF['ARRDELAY'],
                                    100, color='green')
plt.title('Histogram of Arrival Delay')
plt.xlabel('Arrival Delay (minutes)')
plt.ylabel('# of flights')
```

```
oml.graphics.boxplot(IRIS[:,:4], notch=True,
showmeans = True, labels=['Sepal Length',  'Sepal
    Width','Petal Length', 'Petal  Width'])
plt.title('Distribution of IRIS Attributes')
plt.ylabel('cm');
```

オープン・ソース言語によるOracle Databaseへのアクセス

OML4R

```
df1 <-data.frame(x1=1:5,y1=letters[1:5])  df2 <-

data.frame(x1=5:1,y2=letters[11:15])


ore.drop(table="TEST_DF1")
ore.drop(table="TEST_DF2")


ore.create(df1, table="TEST_DF1")
ore.create(df2, table="TEST_DF2")
```

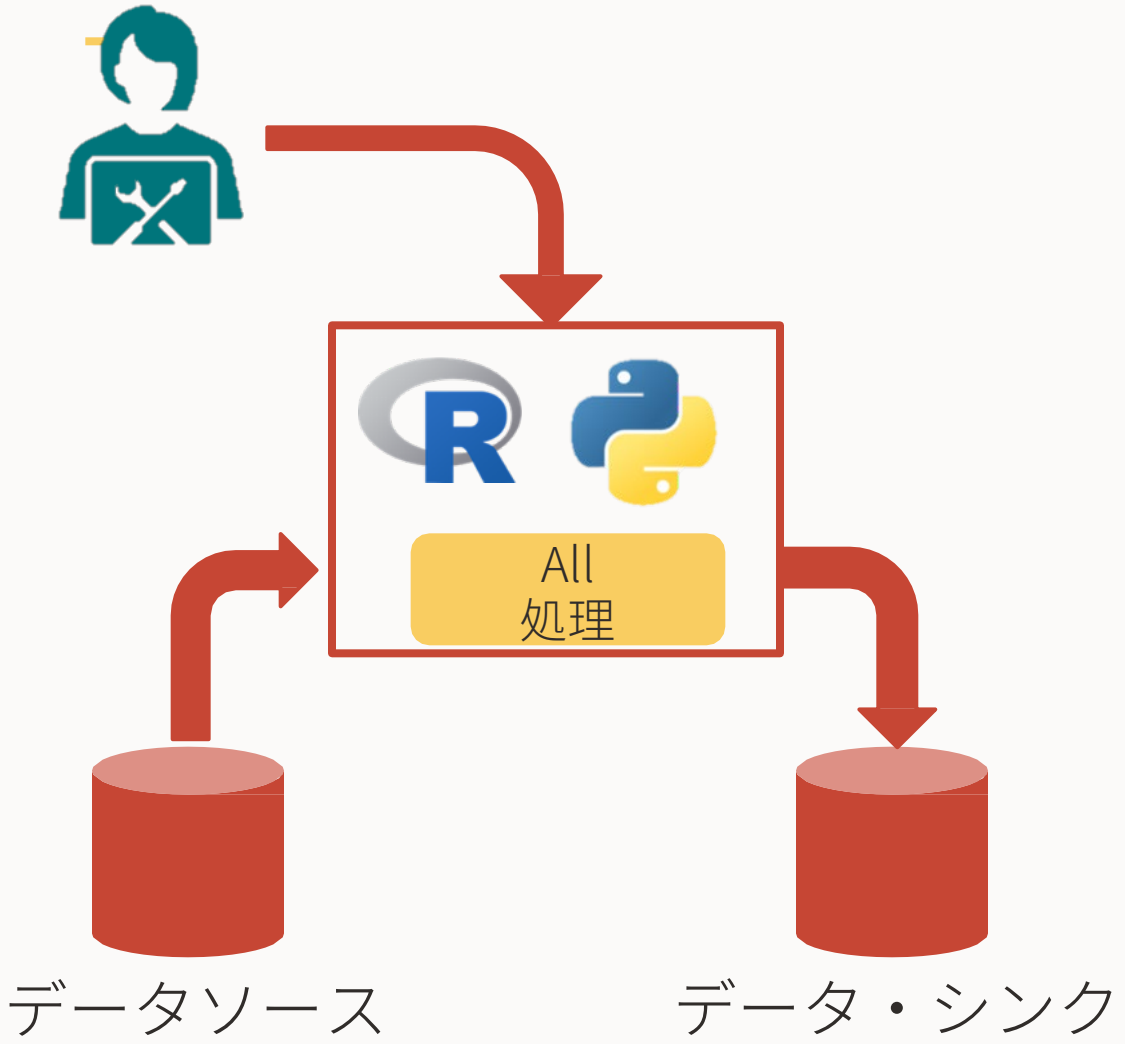
```
merge(TEST_DF1, TEST_DF2, by="x1")
```

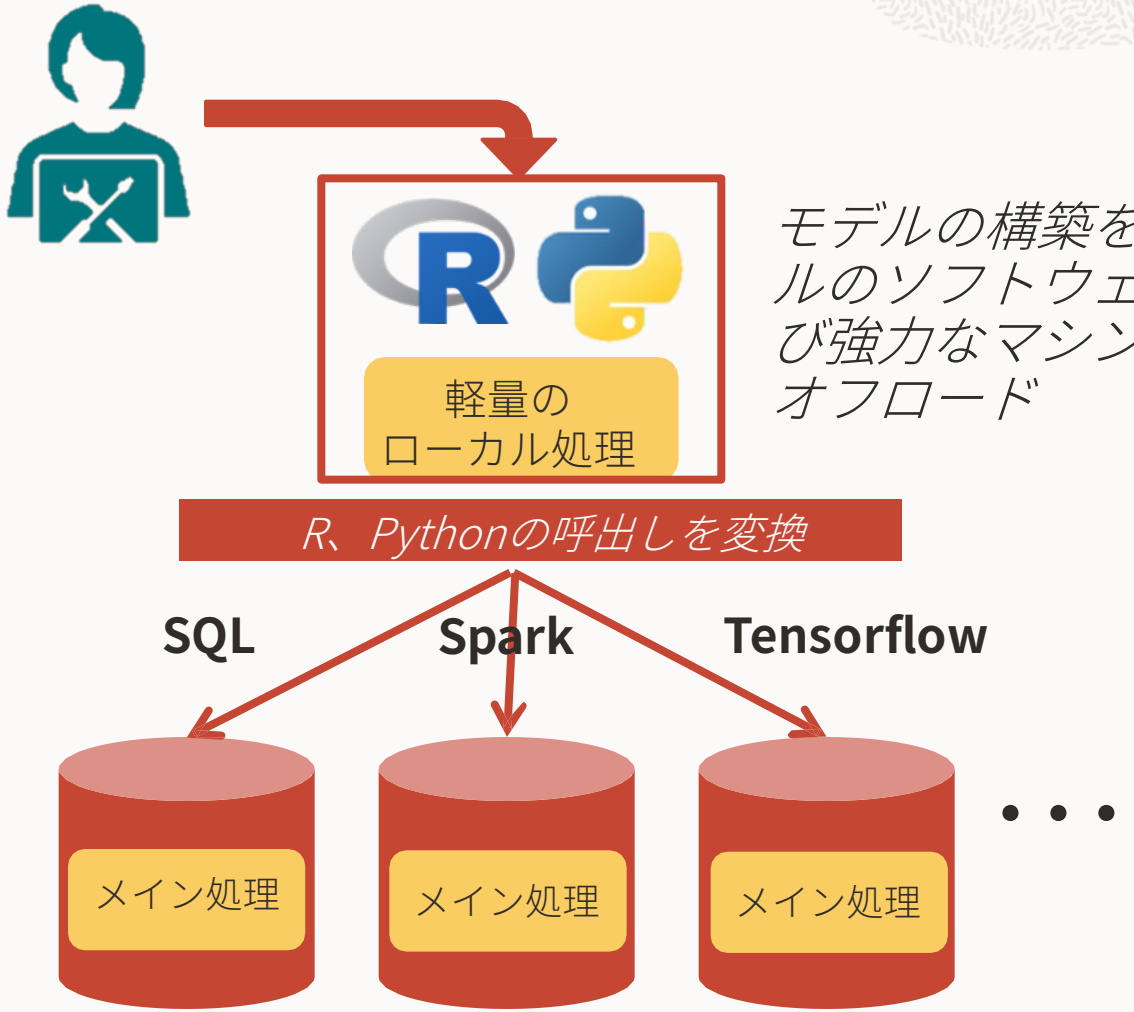
OML4Py*

```
letters = list(map(chr, range(97, 123)))
df1 = pd.DataFrame({'x1':range(1,6),
                    'x2':letters[1:6]})  df2 =
pd.DataFrame({'x1':range(1,6),
                    'x2':letters[11:16]})
oml.drop("TEST_DF1")
oml.drop("TEST_DF2")


TEST_DF1 = oml.create(df1, table="TEST_DF1")
TEST_DF2 = oml.create(df2, table="TEST_DF2")
```

```
df2.merge(df1,on='x1')
```

スケーラブルな機械学習



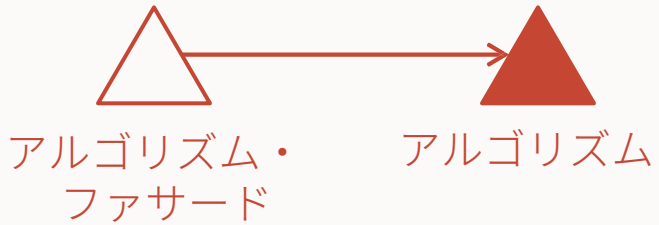


モデルの構築をパラレルのソフトウェアおよび強力なマシン/環境にオフロード

スケーラブルな機械学習

オープン・ソースの機械学習インタフェースをそのまま活用

- OML4R - 記述しやすいR計算式 – コード行数は最小限 変換、インタラクション用の語句など
- OML4Py – 使い慣れたPython予測変数/ターゲット・インタフェース （fit()、predict()を使用）



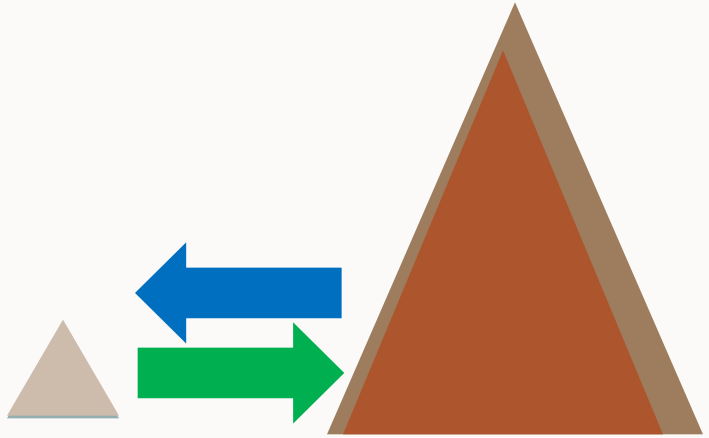
スケーラブルな機械学習

オープン・ソースの機械学習インタフェースをそのまま活用

- OML4R - 記述しやすいR計算式 – コード行数は最小限 変換、インタラクション用の語句など
- OML4Py – 使い慣れたPython予測変数/ターゲット・インタフェース （fit()、predict()を使用）

アルゴリズムをデータの側に配置

- データの移動が不要または最小限に
- プロキシ・オブジェクトを利用して、R/Pythonからデータを参照



スケーラブルな機械学習

オープン・ソースの機械学習インタフェースをそのまま活用

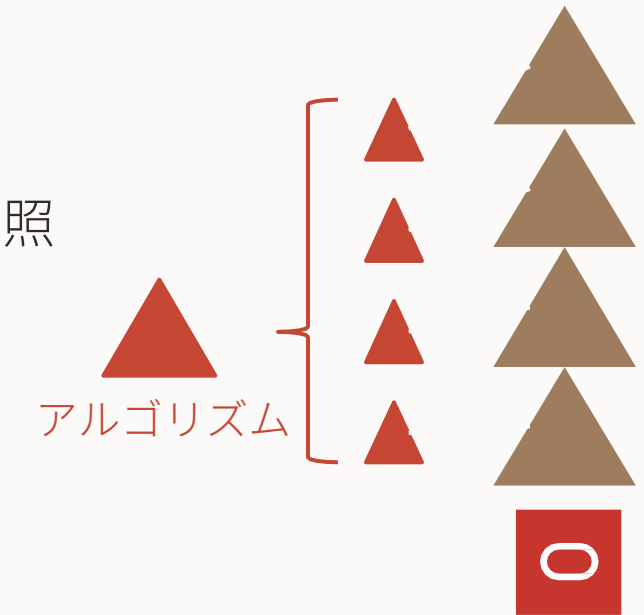
- OML4R - 記述しやすいR計算式 – コード行数は最小限 変換、インタラクション用の語句など
- OML4Py – 使い慣れたPython予測変数/ターゲット・インタフェース （fit()、predict()を使用）

アルゴリズムをデータの側に配置

- データの移動が不要または最小限に
- プロキシ・オブジェクトを利用して、R/Pythonからデータを参照

並列分散アルゴリズムの実装

- カスタム、最先端の統合型実装
- オープン・ソースのパッケージおよびツールキットにより補完



モデルの構築とスコアリングをOracle Databaseにオフロード

OML4R

```
n.rows <- nrow(IRIS)  row.names(IRIS) <-
IRIS$Species
my.smpl <- sample(1:n.rows, ceiling(n.rows*0.7))
train.dat <- IRIS[my.smpl,]
test.dat <- IRIS[setdiff(1:n.rows, my.smpl),]

rf_mod <- ore.randomForest(Species~.,train.dat)
```

```
pred <- predict(rf_mod, test.dat, type="all",
     supplemental.cols=c("SEPAL_LENGTH","Species"))


table(pred$Species, pred$prediction)
```

OML4Py*

```
from oml import rf

train_dat, test_dat = IRIS.split()  train_x =
train_dat.drop('Species')  train_y =
train_dat['Species']

rf_mod = rf()
rf_mod = rf_mod.fit(train_x, train_y)
```

```
pred = rf_mod.predict(test_dat.drop('Species'),
       supplemental_cols = test_dat[:,['SEPAL_LENGTH',
                                       'Species']])
res_ct =
     pred.crosstab('Species','PREDICTION',pivot=True)
res_ct.sort_values(by='Species')
```

パラレル処理とデータ移動排除のメリット

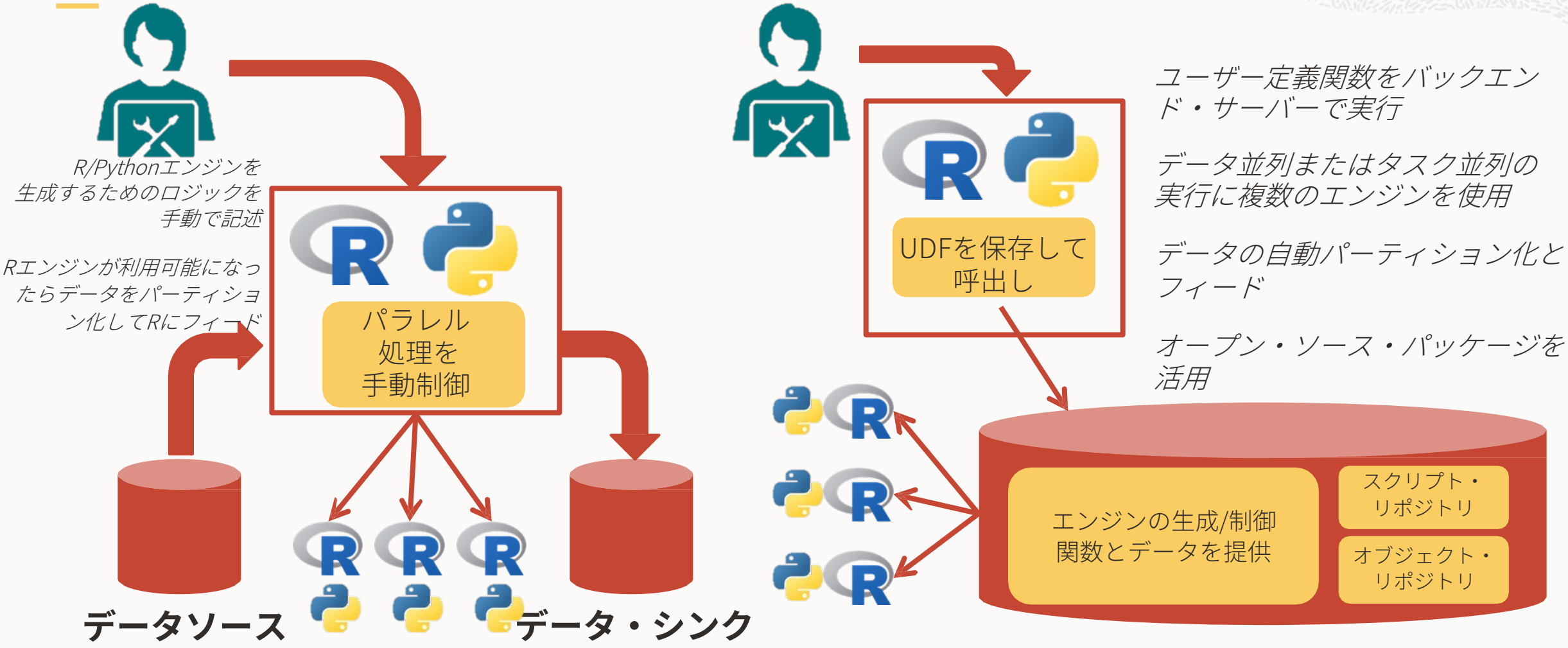
データがOracle Databaseサーバー・マシンに存続

- ある顧客の“収益合計”を予測
- 1億8,400万レコード、31の数値型予測変数
- データはOracle Databaseの表に保存

アルゴリズム	使用スレッド数*	必要メモリ**	データ・ロード時間***	コンピュートのみの 経過時間	経過時間合計	相対的パフォーマンス
オープン・ソースのR線型モデル（lm）	1	220 Gb	1時間3分	43分	1時間46分	1X
OML4R lm（ore.lm）	1	-	-	42.8分	42.8分	2.47X
OML4R lm（ore.lm）	32	-	-	1分34秒	1分34秒	67.7X
OML4R lm（ore.lm）	64	-	-	57.97秒	57.97秒	110X
OML4R lm（ore.lm）	128	-	-	41.69秒	41.69秒	153X

* オープン・ソースのR lm()はシングルスレッドで実行されます
** オープン・ソースのlm()ではすべてのデータをメモリ内に格納する必要があるため、データはR Sessionメモリに移動します
*** 40 GbのRAWデータをオープン・ソースのR Sessionのメモリにロードするためにかかる時間

データおよびタスクのパラレル実行



ユーザー定義関数をバックエンド・サーバーで実行

データ並列またはタスク並列の実行に複数のエンジンを使用

データの自動パーティション化とフィード

オープン・ソース・パッケージを活用

データおよびタスクのパラレル実行

パラレル処理およびデータのパーティション化の容易な指定

- 簡素化されたAPI – オールインワン
- 大量のオープン・ソース・モデルを使用した構築、スコアリング
- 価値または個数によるデータのパーティション化
- ユーザー定義関数を索引入力とパラレルで呼出し

R、Pythonエンジンの自動管理

- ハードウェアの詳細から分離
- 可能な範囲でメモリ・リソースとコンピュート・リソースを制限

データとユーザー定義関数をRエンジン、Pythonエンジンに自動ロード

オープン・ソース・エコシステムのパッケージを活用

OML4RとOML4Pyの比較 – 表の適用（Table Apply）

OML4R

```
library(e1071)
buildNB <-
    function(dat,dsname){
        library(e1071)
        dat$Species <- as.factor(dat$Species)
        mod<-naiveBayes(Species ~ ., dat)
        ore.save(mod,name=dsname,overwrite=TRUE)  mod
    }
```

```
mod <- ore.tableApply(IRIS, buildNB,
                                    dsname='NB_Model-1',
                                    ore.connect=TRUE)
```

OML4Py*

```
def build_nb(dat, dsname):
        import oml
        from sklearn.naive_bayes import GaussianNB
        from sklearn import preprocessing
        le = preprocessing.LabelEncoder()
        raw_labels = dat[["Species"]].values
        le.fit(raw_labels)
        y = le.transform(raw_labels)
        X = dat[["SEPAL_LENGTH","SEPAL_WIDTH",
        "PETAL_LENGTH","PETAL_WIDTH"]].values
        mod = GaussianNB().fit(X, y)
        oml.ds.save(objs={'mod':mod},name=dsname,
                                overwrite=True)
```

```
mod = oml.table_apply(IRIS, build_nb,
                                    dsname = 'NB_Model-1',
                                    oml_connect=True)
```

OML4RとOML4Pyの比較 – 行の適用（Row Apply）

OML4R

```
scoreNBmodel <- function(dat, dsname) {
        library(e1071)
        ore.load(dsname)
        dat$PRED <- predict(mod, newdata = dat)
        dat
    }
```

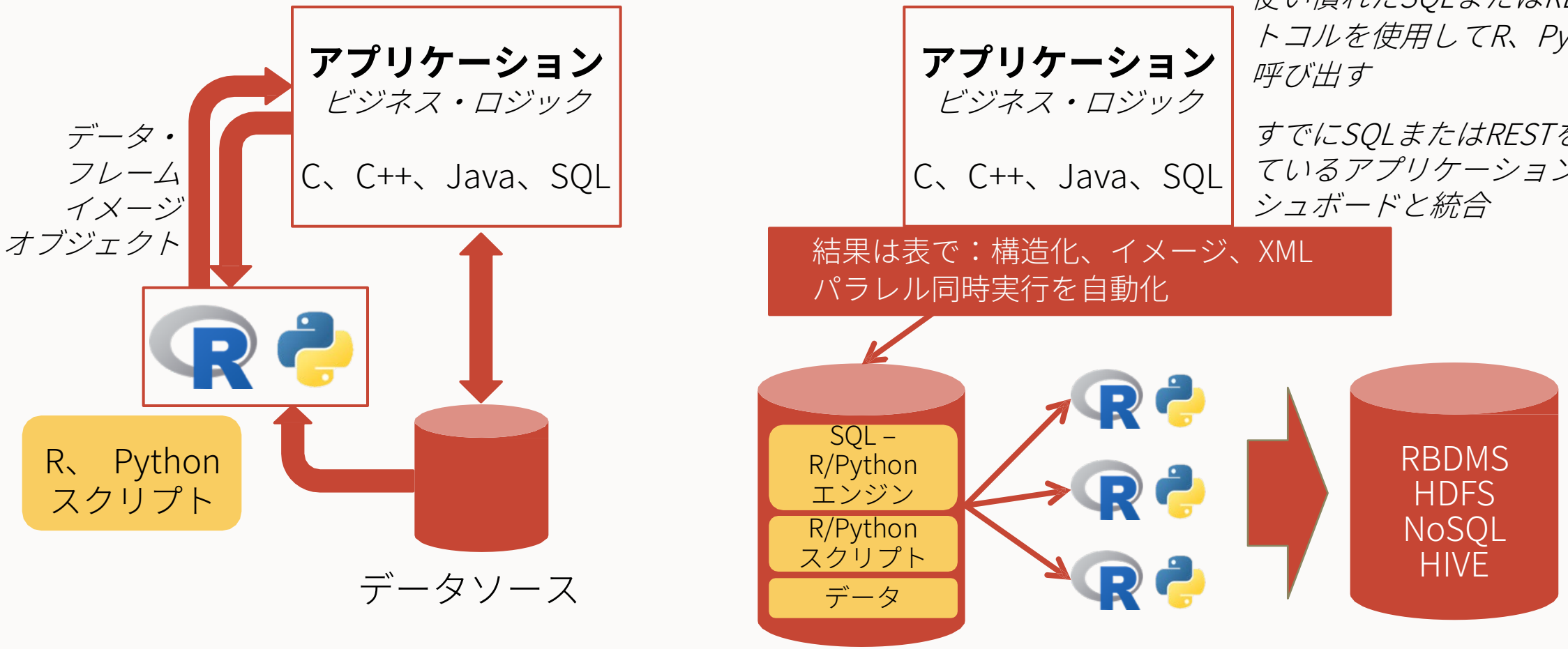
```
IRIS_PRED           <- IRIS[1,]
IRIS_PRED$PRED <- "A"

res <- ore.rowApply(IRIS, scoreNBmodel,
                                dsname = 'NB_Model-1',
                                parallel = 4, rows = 10,
                                FUN.VALUE = IRIS_PRED,
                                ore.connect = TRUE)
```

OML4Py*

```
def score_nb_mod(dat, dsname):
        import oml
        from sklearn.naive_bayes import GaussianNB
        objs = oml.ds.load(dsname,to_globals=False)
        mod = objs['mod']
         dat['PREDICTION'] =
                mod.predict(dat.drop('Species',axis=1))
        return dat
```

```
IRIS_PRED = pd.DataFrame([(1,1,1,1,'a',1)],
      columns=["SEPAL_LENGTH","SEPAL_WIDTH",
                        "PETAL_LENGTH","PETAL_WIDTH",
                        "Species","PREDICTION"]))
res = oml.row_apply(IRIS,score_nb_mod,
                                dsname = 'NB_Model-1',
                                parallel = 4, rows = 10
                                func_value = IRIS_PRED,
                                oml_connect = True)
```

デプロイメント



デプロイメント

- すでにSQLまたはRESTを使用している環境からユーザー定義のRおよびPythonの関数を容易に呼出し可能
- データの構造と型の自動マッピング
- データ・フレーム、イメージ、XMLをデータベース行セットとしてシームレスに返却
- 完全自動操作のために、ユーザー定義関数の実行スジュールを設定

SQLからのユーザー定義関数の作成（またはR/Pythonからの使用）

OML4R

```
BEGIN
    sys.rqScriptDrop('RandomRedDots');
  sys.rqScriptCreate('RandomRedDots',
  'function(){
            id <- 1:10
            plot( 1:100, rnorm(100), pch = 21,
            bg = "red", cex = 2,  main="Random Red
                    Dots" )
            data.frame(id=id, val=id / 100)
            }');
END;
```

OML4Py*

```
BEGIN
      sys.pyqScriptDrop('RandomRedDots');
      sys.pyqScriptCreate('RandomRedDots',
'def RandomRedDots ():
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

        d = {''id'': range(1,10),
                ''val'': [x/100 for x in range(1,10)]}
        df = pd.DataFrame(data=d)
        fig = plt.figure(1)
        ax = fig.add_subplot(111)
        ax.scatter(range(0,100),
                        np.random.rand(100),c=''r'')
        fig.suptitle("Random Red Dots")
        return df', NULL, TRUE);
        END;
```

SQLからのユーザー定義関数の呼出し

OML4R

```
select ID, IMAGE from
    table(rqEval(NULL,'PNG','RandomRedDots'))
```

```
select ID, VAL from
    table(rqEval(NULL,
                 'select 1 id, 1 val from dual',
                 'RandomRedDots'))
```

```
select dbms_lob.substr(VALUE,4000,1) from
    table(rqEval(NULL,'XML',RandomRedDots'))
```

```
# In R, invoke same function by name
ore.doEval(FUN.NAME='RandomRedDots')
```

OML4Py*

```
select ID, IMAGE from
    table(pyqEval(NULL,'PNG','RandomRedDots'))
```

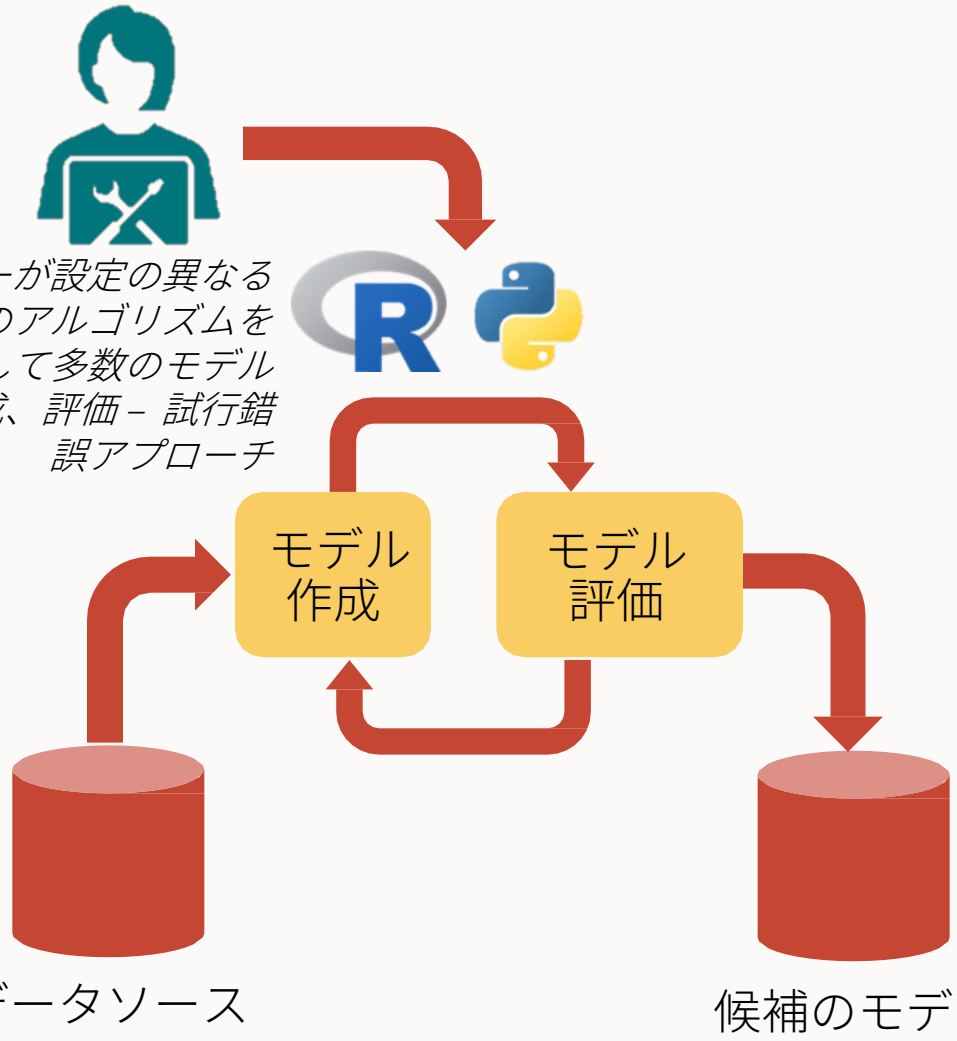
```
select ID, VAL from
    table(pyqEval(NULL,
                 'select 1 id, 1 val from dual',
                 'RandomRedDots'))
```

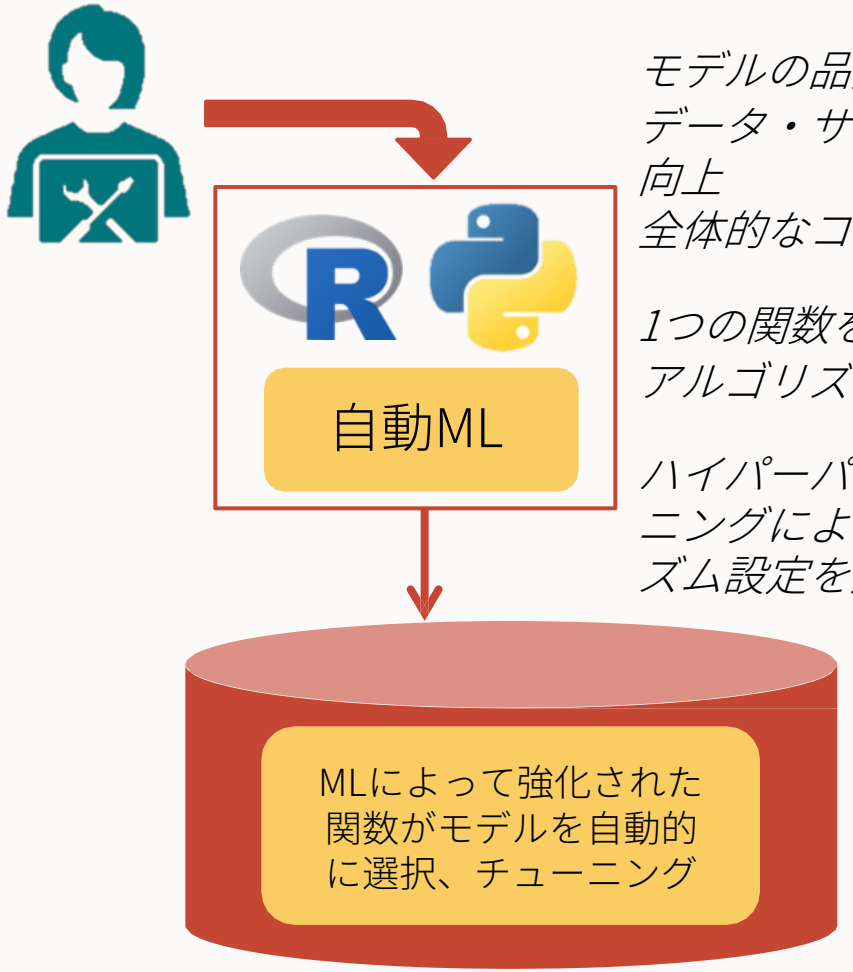
```
select dbms_lob.substr(VALUE,4000,1) from
    table(pyqEval(NULL,'XML','RandomRedDots'))
```

```
# In Python, invoke same function by name
res = oml.do_eval(func='RandomRedDots')
```

自動化

ユーザーが設定の異なる複数のアルゴリズムを使用して多数のモデルを作成、評価 – 試行錯誤アプローチ





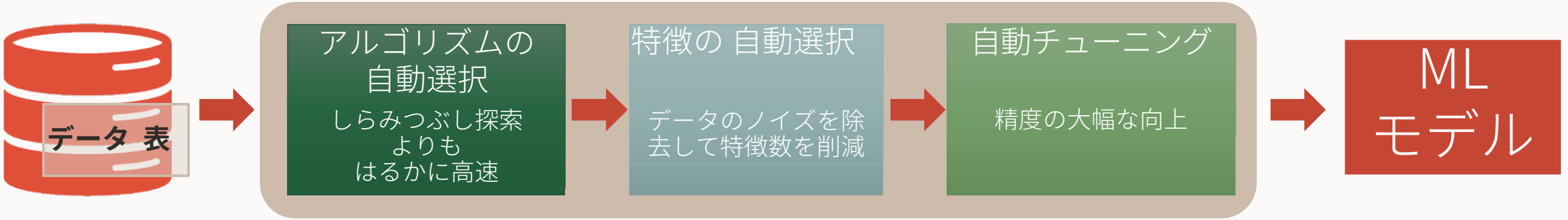
モデルの品質の向上
データ・サイエンティストの生産性向上
全体的なコンピュート時間を削減

1つの関数を呼び出して“最適な”アルゴリズムとモデルを検索

ハイパーパラメータの自動チューニングによって“最適な”アルゴリズム設定を選択

AutoML – OML4Pyの*新機能*

データ・サイエンティストの生産性向上 – 全体的なコンピュート時間を削減



アルゴリズムの自動選択

- もっとも高い品質のモデルとなるインデータベース・アルゴリズムを特定
- しらみつぶし探索よりも高速に実行できる最適なモデルを検索

特徴の自動選択

- もっとも良い予測を特定して特徴数を削減
- パフォーマンスと精度の向上

ハイパーパラメータを自動でチューニング

- モデル精度が大幅に向上
- 手動探索またはしらみつぶし探索の手法を回避

エキスパートでないユーザーでも機械学習の活用が可能に

アルゴリズムの自動選択

MLが導く最大のスコア指標を達成するための最適なアルゴリズムを特定
しらみつぶし探索よりも数倍高速に実行できる最適なモデルを検索

OML4Py

```
ms = AlgorithmSelection(mining_function = 'classification',
                                        score_metric = 'accuracy',
                                        parallel = 4)

best_model = ms.select(X_train, y_train)

all_models = ms.select(X_train, y_train, tune=False)
```

特徴の自動選択

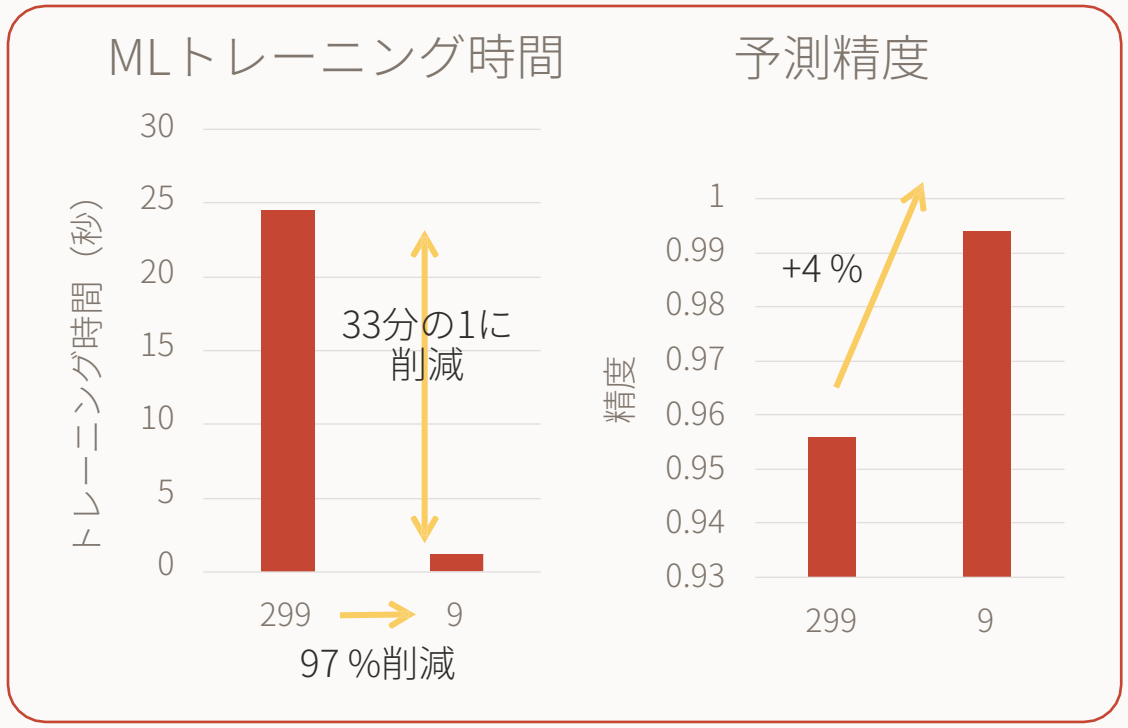
もっとも関連のある特徴を選択することでMLパイプラインを高速化

OML4Py

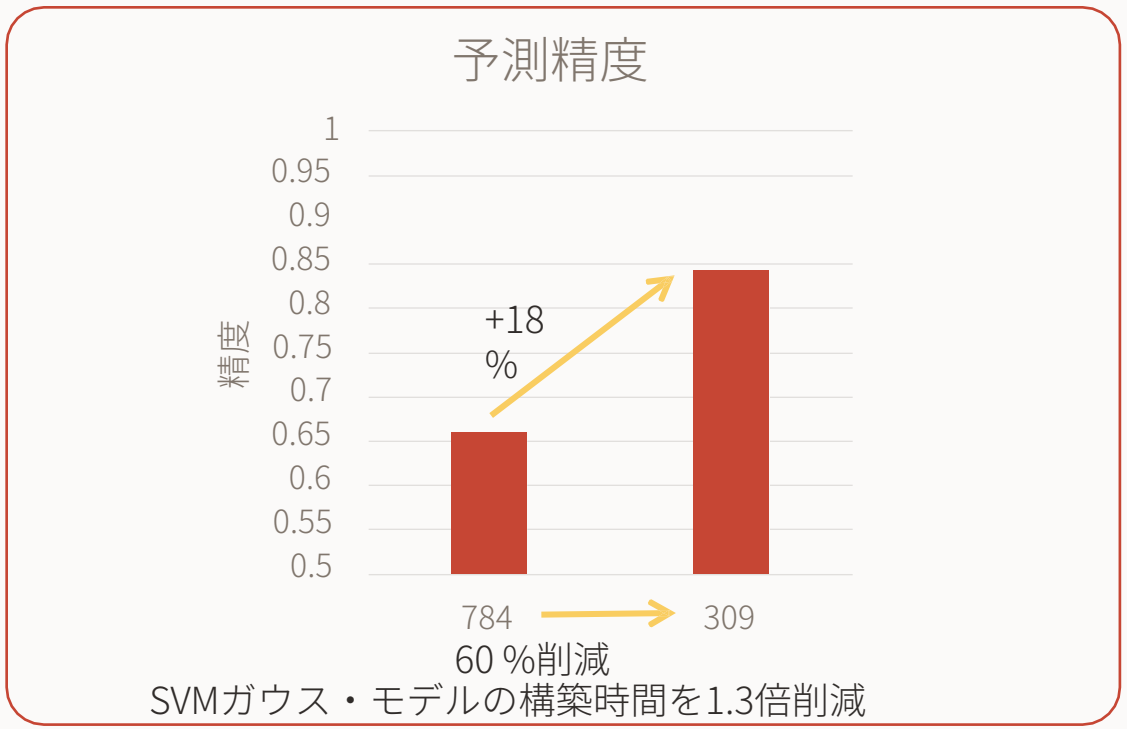
```
fs = FeatureSelection(mining_function = 'classification',
                      score_metric = 'accuracy',
                      parallel = 4)

selected_features = fs.reduce('dt', X_train, y_train)

X_train = X_train[:,selected_features]
```

OML4Pyによる特徴の自動選択：事例

もっとも高い関連を特定することで特徴数を削減し、パフォーマンスと精度を向上



OpenMLのデータセット312（1925行、299列）



OpenMLのデータセット40996（56000行、784列）

自動チューニング

MLが導くモデル精度が大幅に向上

手動探索またはしらみつぶし探索の手法を回避

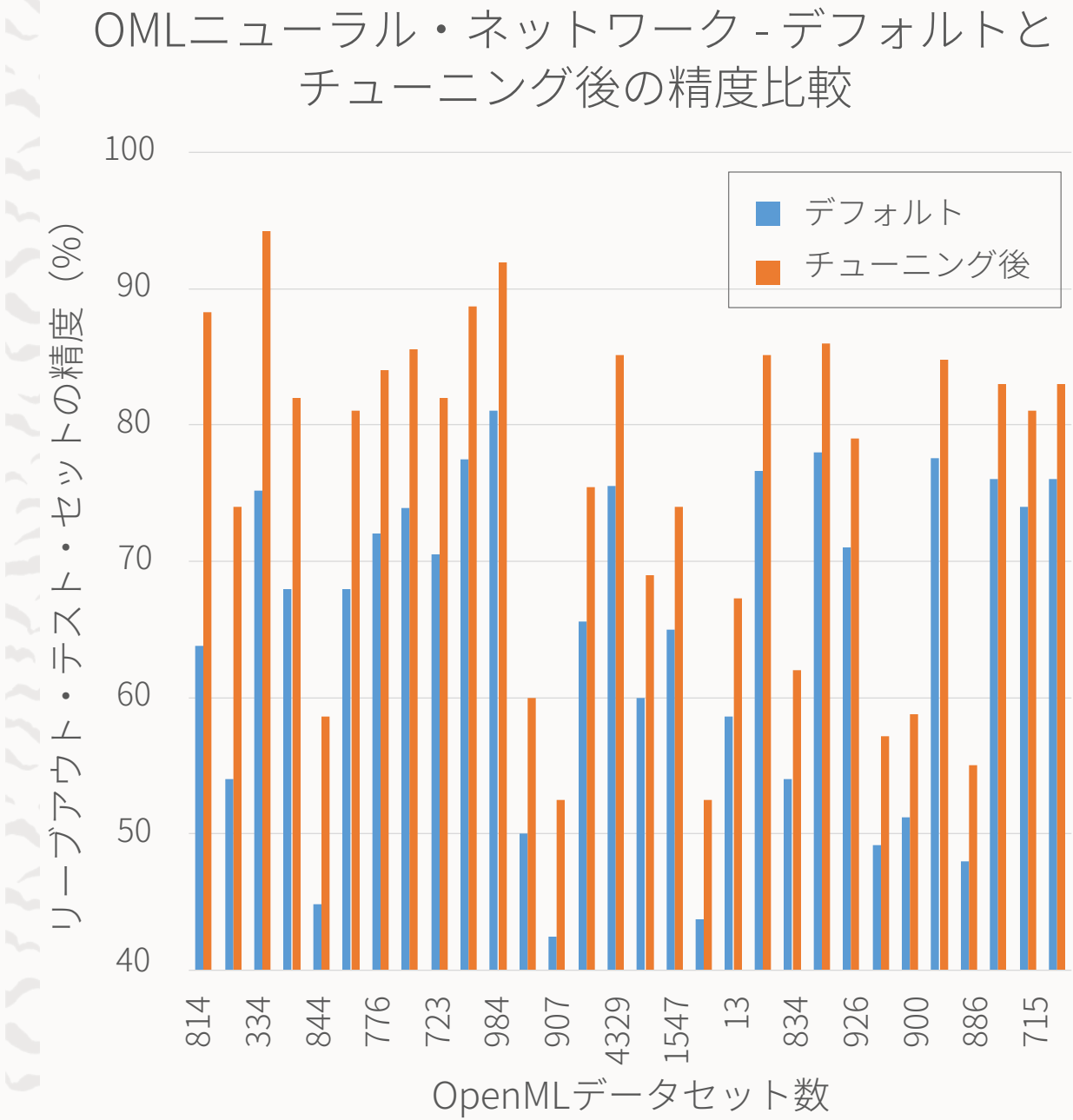
OML4Py

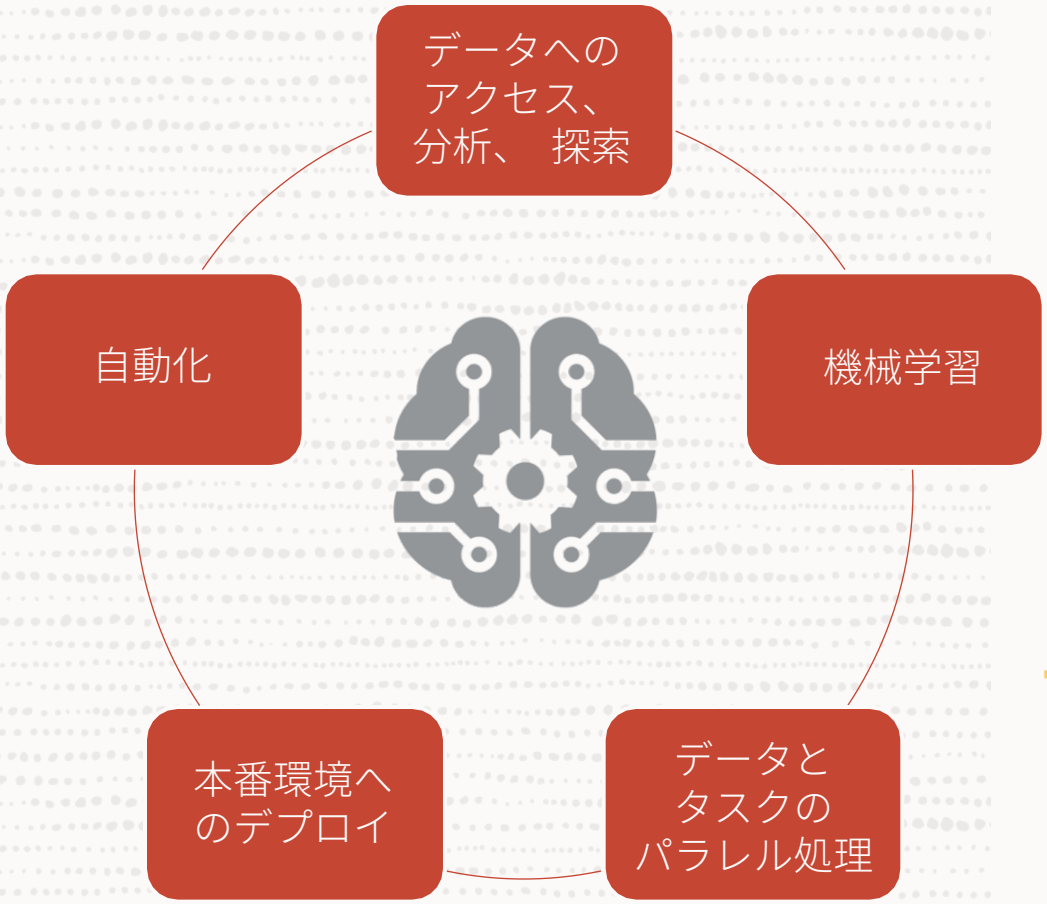
```
at = Autotune(mining_function = 'classification',
                    score_metric = 'accuracy',
                        parallel = 4)

results = at.tune('dt', X_train, y_train)

best_mod = results['best_model']
all_evals = results['all_evals']
```

約300個のデータセット全体で平均1.7％改善

一部のデータセットについては 8～24％の改善

自動チューニング：
OMLニューラル・ネットワークの評価





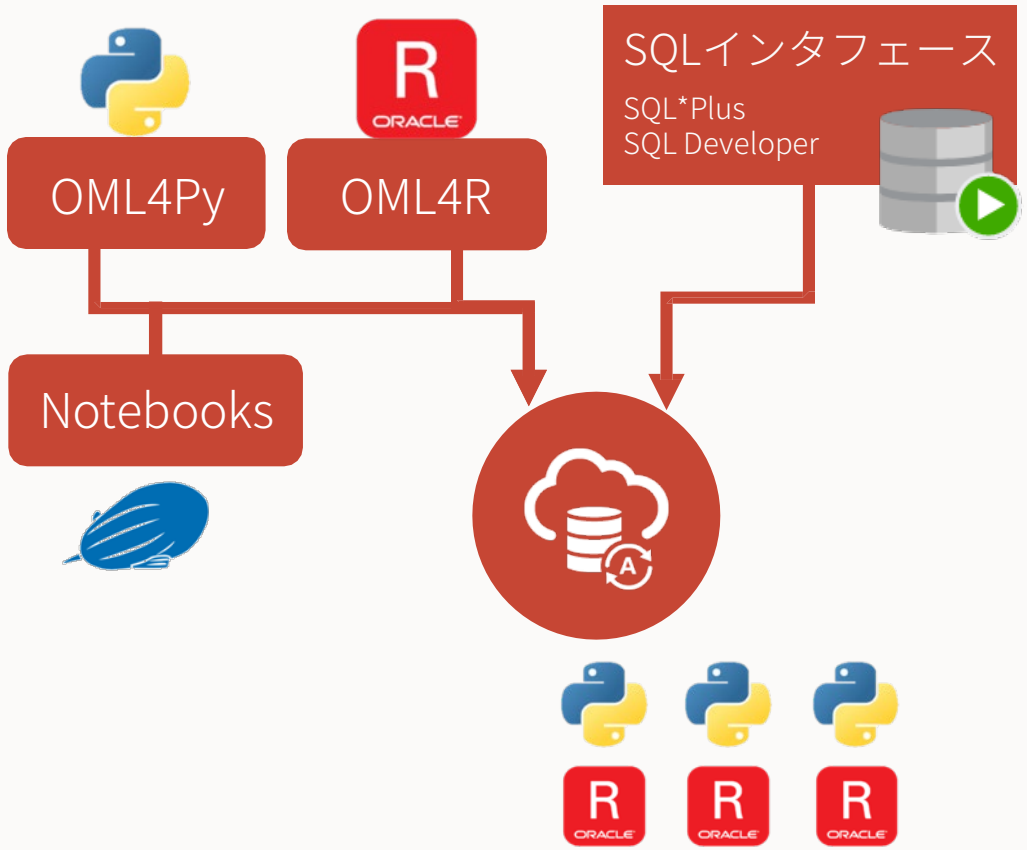
企業でも RおよびPythonが使用可能に

1. 新しくより強力なMLバックエンドおよびライブラリを、そのリリース時にもっと容易に活用可能
2. ビッグ・データの処理のためのデータ並列、タスク並列の実行がより高速かつ容易に
3. 手間のかかる部分について、処理を強力なバックエンドにオフロード - 透過的なスケーラビリティ
4. データ・サイエンティストのR、Pythonのスクリプトと結果を本番環境で即座に活用
5. 自動化によってユーザーの生産性を強化

Oracle Machine Learning for R / Python

Oracle Autonomous Databaseに近日追加予定

- Oracle DatabaseをHPC環境として使用
- インデータベース、並列分散の機械学習アルゴリズムを使用
- Oracle Database内でR、Pythonのスクリプトとオブジェクトを管理
- オープン・ソースの結果をSQL経由でアプリケーションおよびダッシュボードに統合
- OML4Pyでは、自動機械学習のAutoMLを利用可能



Oracle Machine Learning for R / Python

Oracle Autonomous Databaseに近日追加予定

透過レイヤー

- プロキシ・オブジェクトを活用してデータベース内にデータを保持
- 機能をSQLに変換するネイティブ関数をオーバーロード
- 使い慣れたR/Python構文を使用してデータベース・データを操作

並列分散アルゴリズム

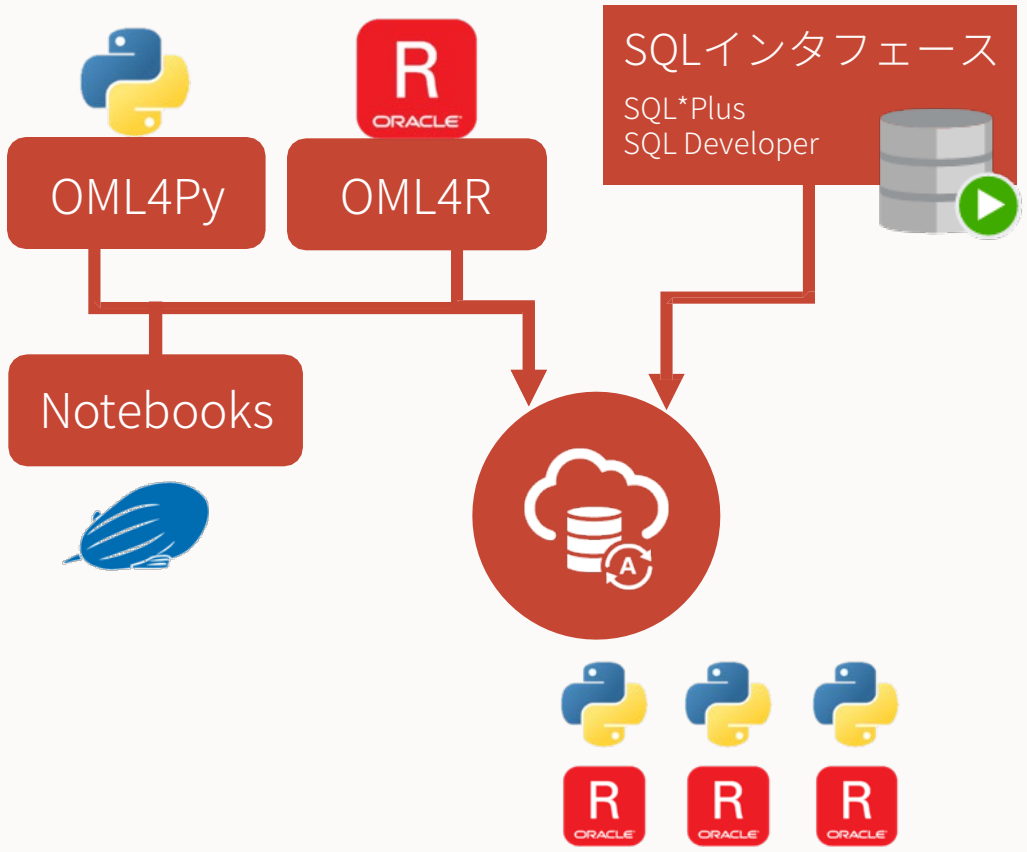
- スケーラビリティとパフォーマンス
- 利用可能なインデータベース・アルゴリズムがOML4SQLにより公開

組込み実行

- Oracle DatabaseでRスクリプトまたはPythonスクリプトを管理して起動
- データ並列、タスク並列、および非パラレルの実行
- オープン・ソースのパッケージを使用して機能を補強

OML4Pyの自動機械学習 - AutoML

- モデルの選択、特徴の選択、ハイパーパラメータ・チューニング



追加情報

oracle.com/machine-learning

AskTOM OML Office Hoursもご覧ください

ORACLE Ask TOM

Ask The Oracle Masters

Oracle Cloud Infrastructure

New Free Tier with Always Free Oracle Autonomous Database and Oracle APEX

oracle.com/cloud/free

Always Free

Services you can use for unlimited time

+

30-Day Free Trial

Get $500 in free credits

LEARN MORE

Oracle Cloud's New Free Tier and Always Free Oracle Autonomous Database

Speaker: Todd Bottger
Session ID: PRO6695
Wednesday, Sept 18th at 10:00am PT
Moscone South #203

Mark Hornick
Marcos Arancibia