

# Predictive models design

Preparation



# Predictive models design

Preparation



Investigation



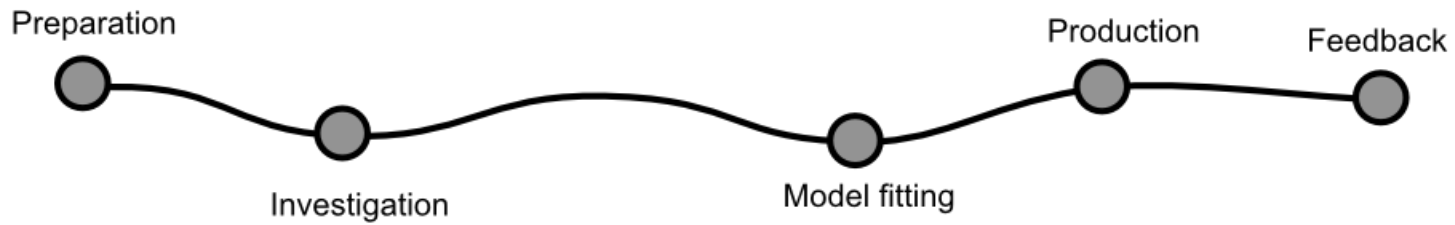
# Predictive models design



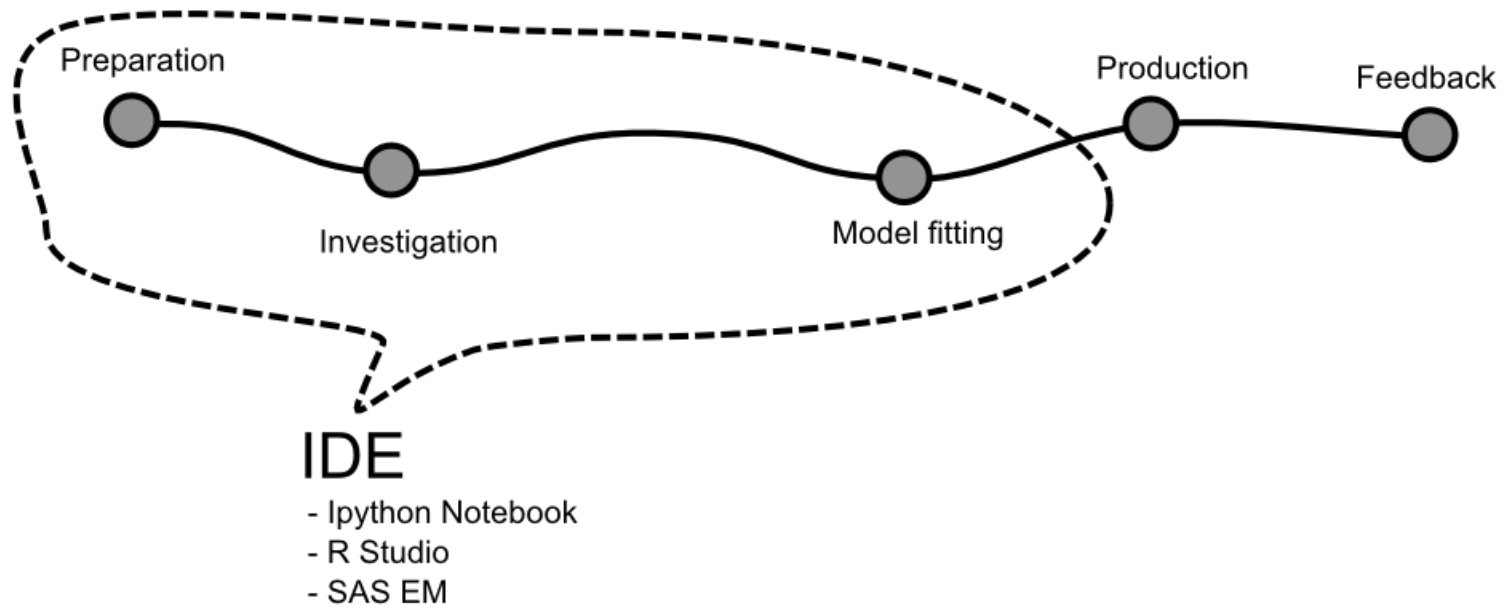
# Predictive models design



# Predictive models design



# Small data case



# IP[y]: Notebook iris example (unsaved changes)

File

Edit

View

Insert

Cell

Kernel

Help



Code



Cell Toolbar:

None



```
In [1]: from sklearn import datasets  
iris = datasets.load_iris()
```

# IP[y]: Notebook iris example (autosaved)

File Edit View Insert Cell Kernel Help

         Code Cell Toolbar: None

```
In [1]: from sklearn import datasets
iris = datasets.load_iris()
```

```
In [ ]: import pandas as pd
pd.set_option('display.max_rows', 10)
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df["target"] = iris.target
iris_df
```



# IP[y]: Notebook iris example (unsaved changes)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
In [1]: from sklearn import datasets
iris = datasets.load_iris()
```

```
In [2]: import pandas as pd
pd.set_option('display.max_rows', 10)
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df["target"] = iris.target
iris_df
```

```
Out[2]:
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows x 5 columns

## IP[y]: Notebook iris example (unsaved changes)

File Edit View Insert Cell Kernel Help

          Code Cell Toolbar: None

```
In [ ]: from sklearn import datasets  
iris = datasets.load_iris()
```

```
In [ ]: import pandas as pd  
pd.set_option('display.max_rows', 10)  
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)  
iris_df["target"] = iris.target  
iris_df
```

```
In [ ]: %matplotlib inline  
import seaborn as sns  
sns.pairplot(iris_df, hue="target", size=2.5)
```

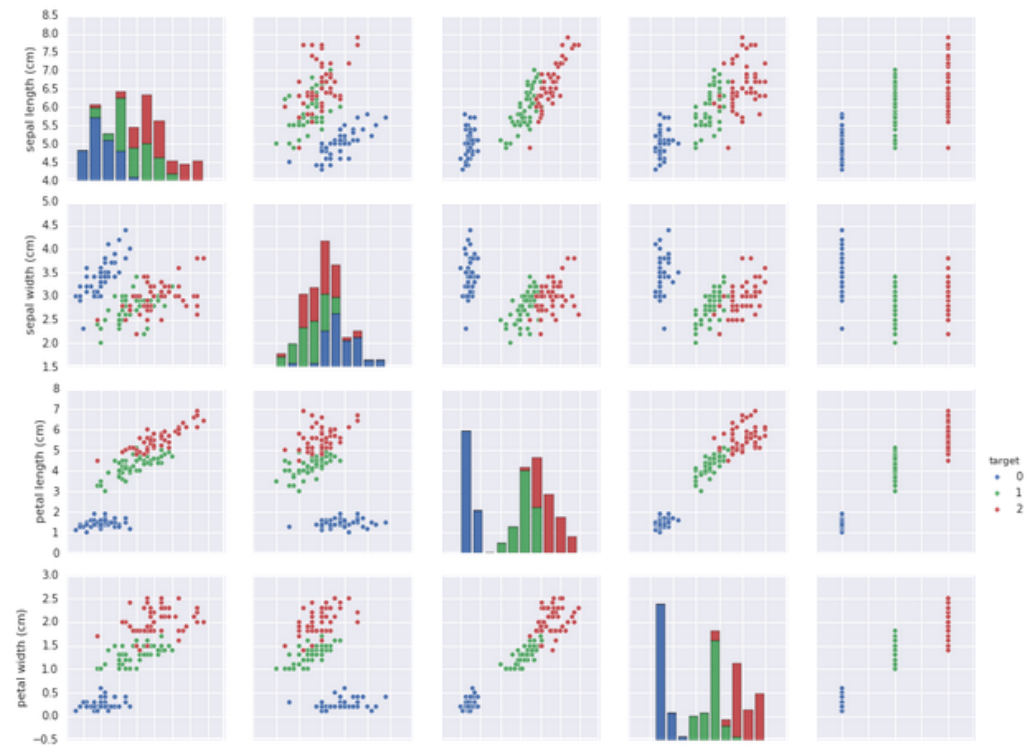
## IP[y]: Notebook iris example (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
In [4]: %matplotlib inline
import seaborn as sns
sns.pairplot(iris_df, hue="target", size=2.5)
```

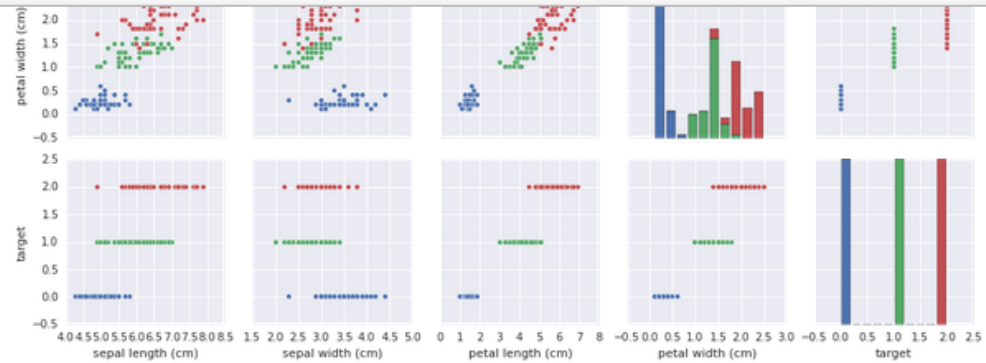
Out[4]: <seaborn.axisgrid.PairGrid at 0x7fc3484a4e10>



# IP[y]: Notebook iris example (unsaved changes)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None



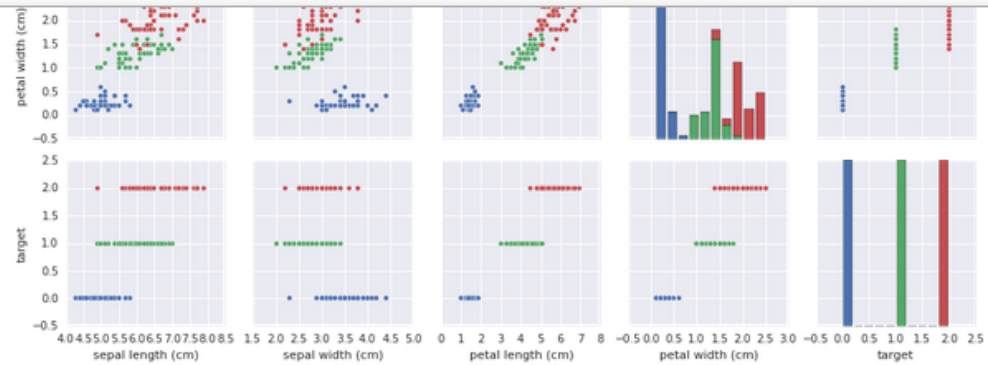
```
In [ ]: from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
model = lr.fit(iris.data, iris.target)

predictions = model.predict(iris.data)
```

# IP[y]: Notebook iris example (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None



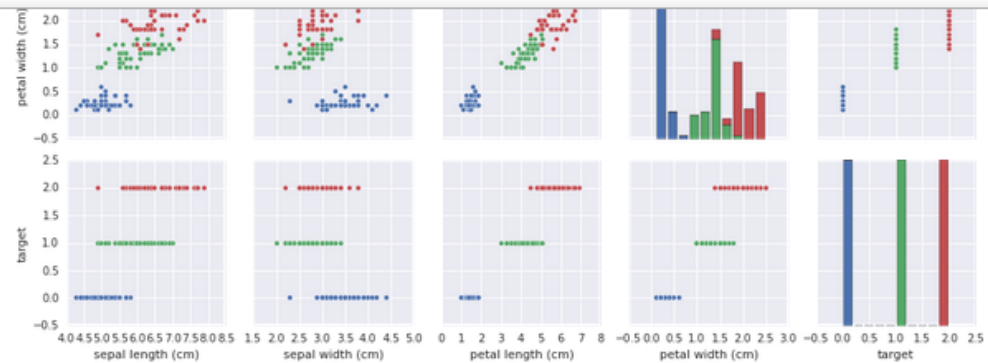
```
In [6]: from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
model = lr.fit(iris.data, iris.target)

predictions = model.predict(iris.data)
```

# IP[y]: Notebook iris example (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None



```
In [6]: from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
model = lr.fit(iris.data, iris.target)

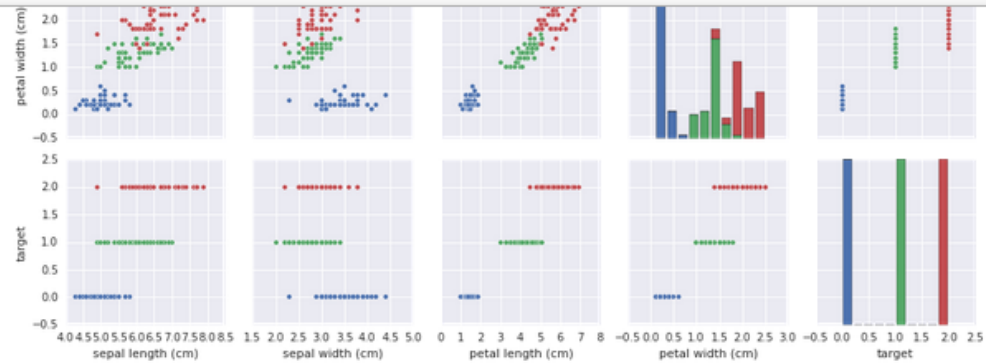
predictions = model.predict(iris.data)
```

```
In [ ]: from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(iris.target, predictions)
pd.DataFrame(confusion, columns=iris.target_names, index=iris.target_names)
```

# IP[y]: Notebook iris example (unsaved changes)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None



```
In [6]: from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
model = lr.fit(iris.data, iris.target)

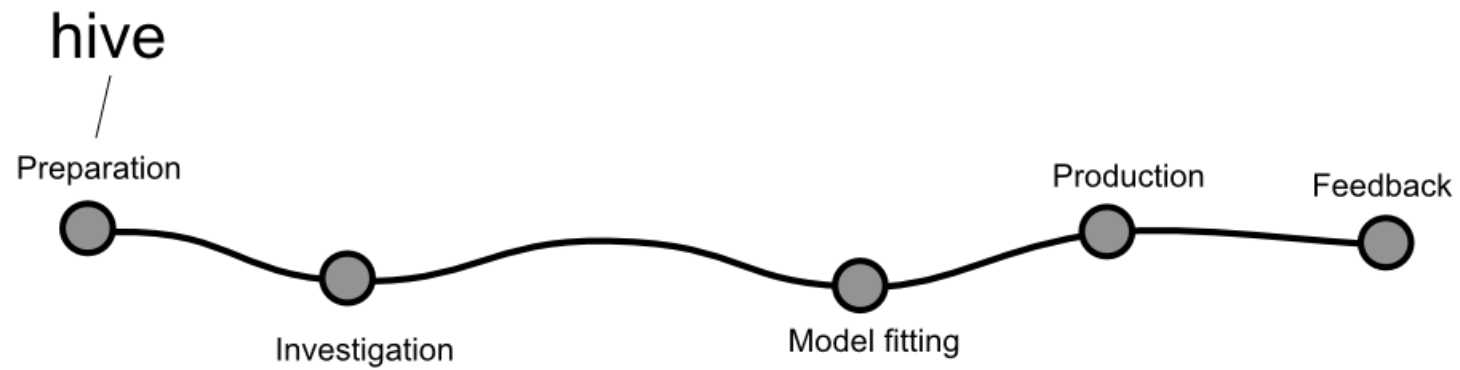
predictions = model.predict(iris.data)
```

```
In [7]: from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(iris.target, predictions)
pd.DataFrame(confusion, columns=iris.target_names, index=iris.target_names)
```

Out[7]:

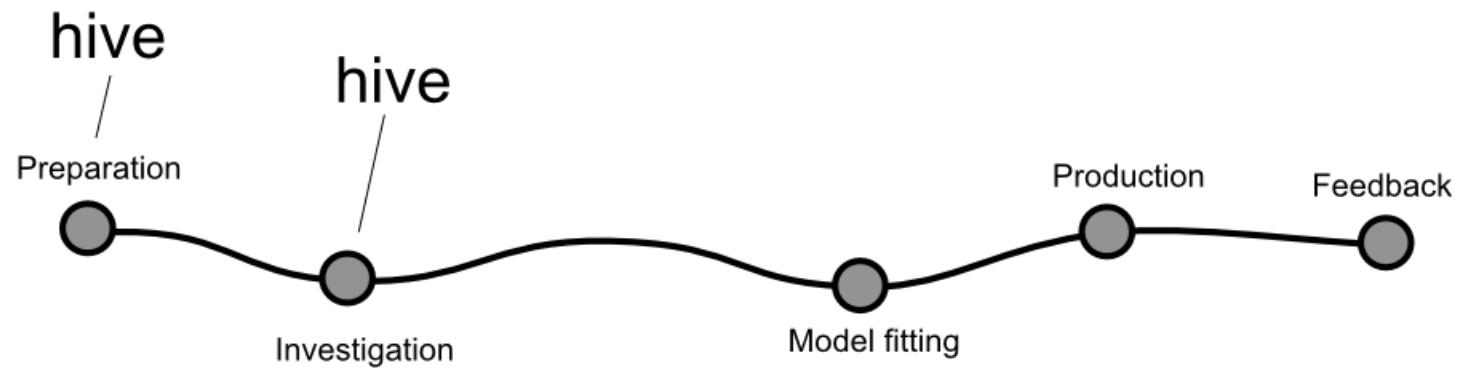
	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	45	5
virginica	0	1	49

# Big data case

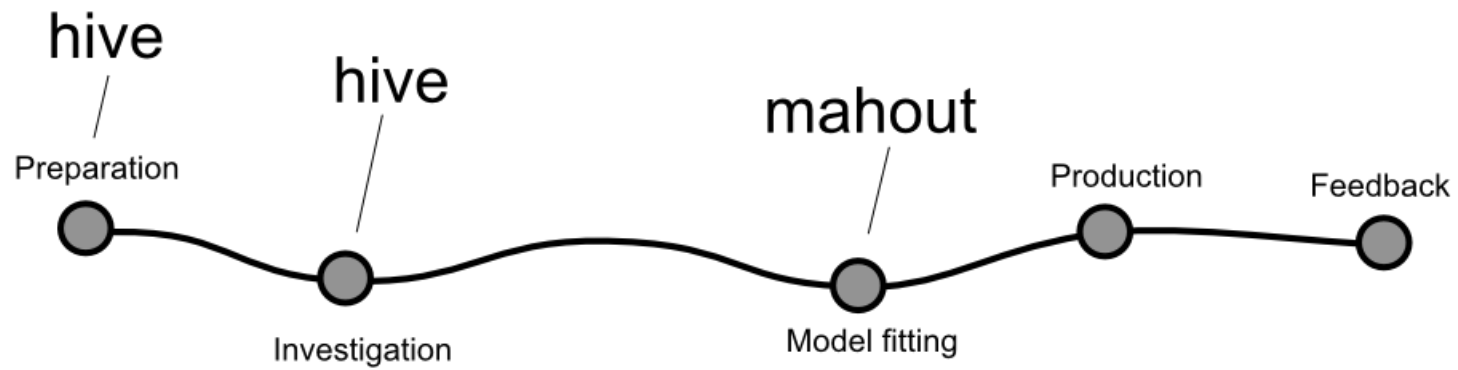




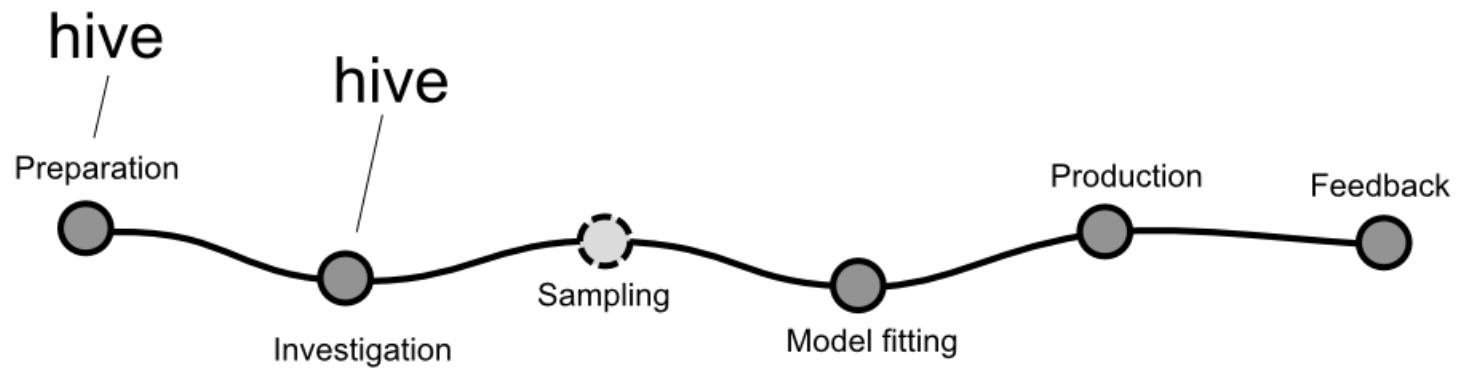
# Big data case



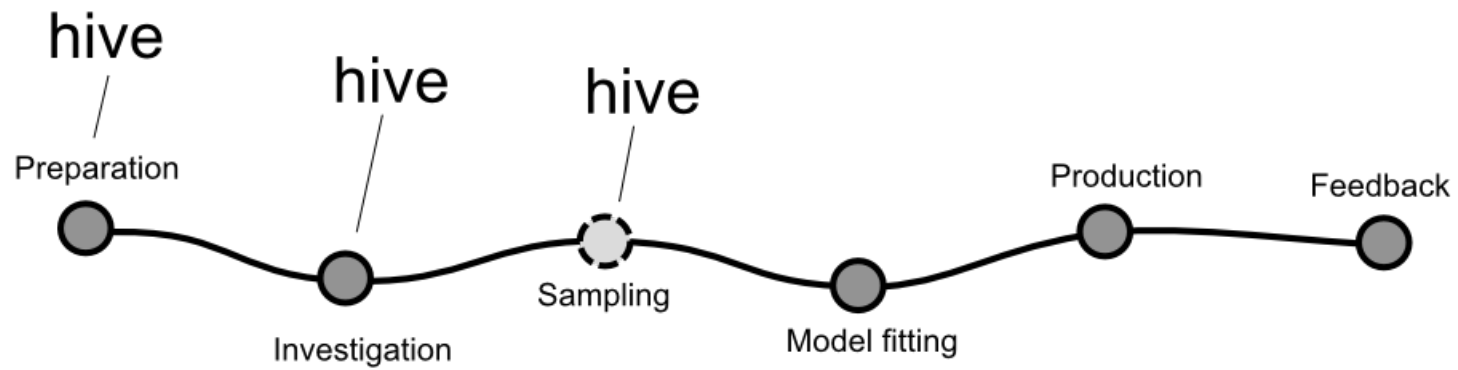
# Big data case



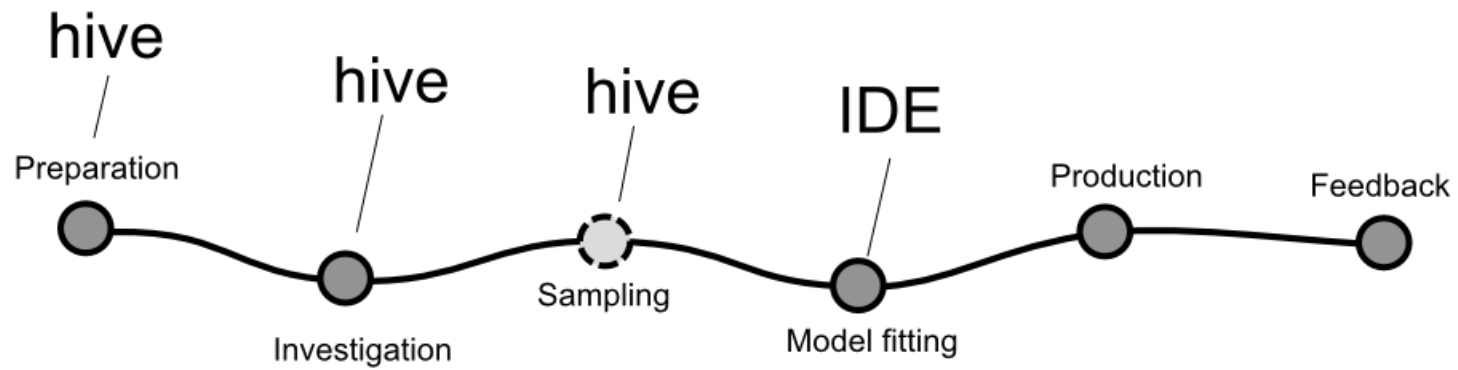
# Big data case



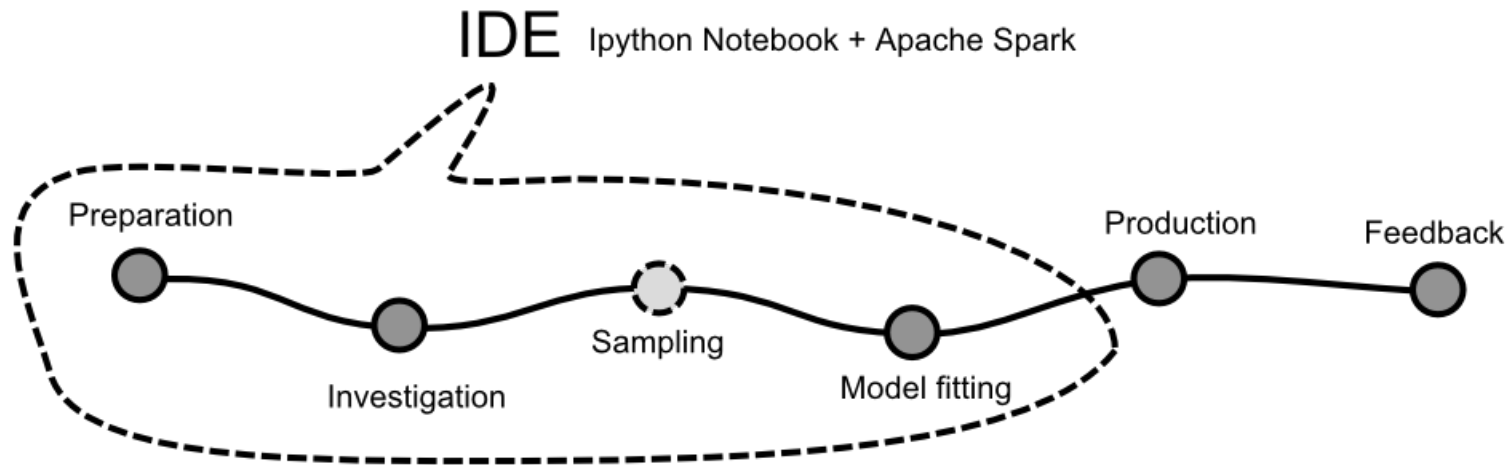
# Big data case



# Big data case



# Apache Spark case

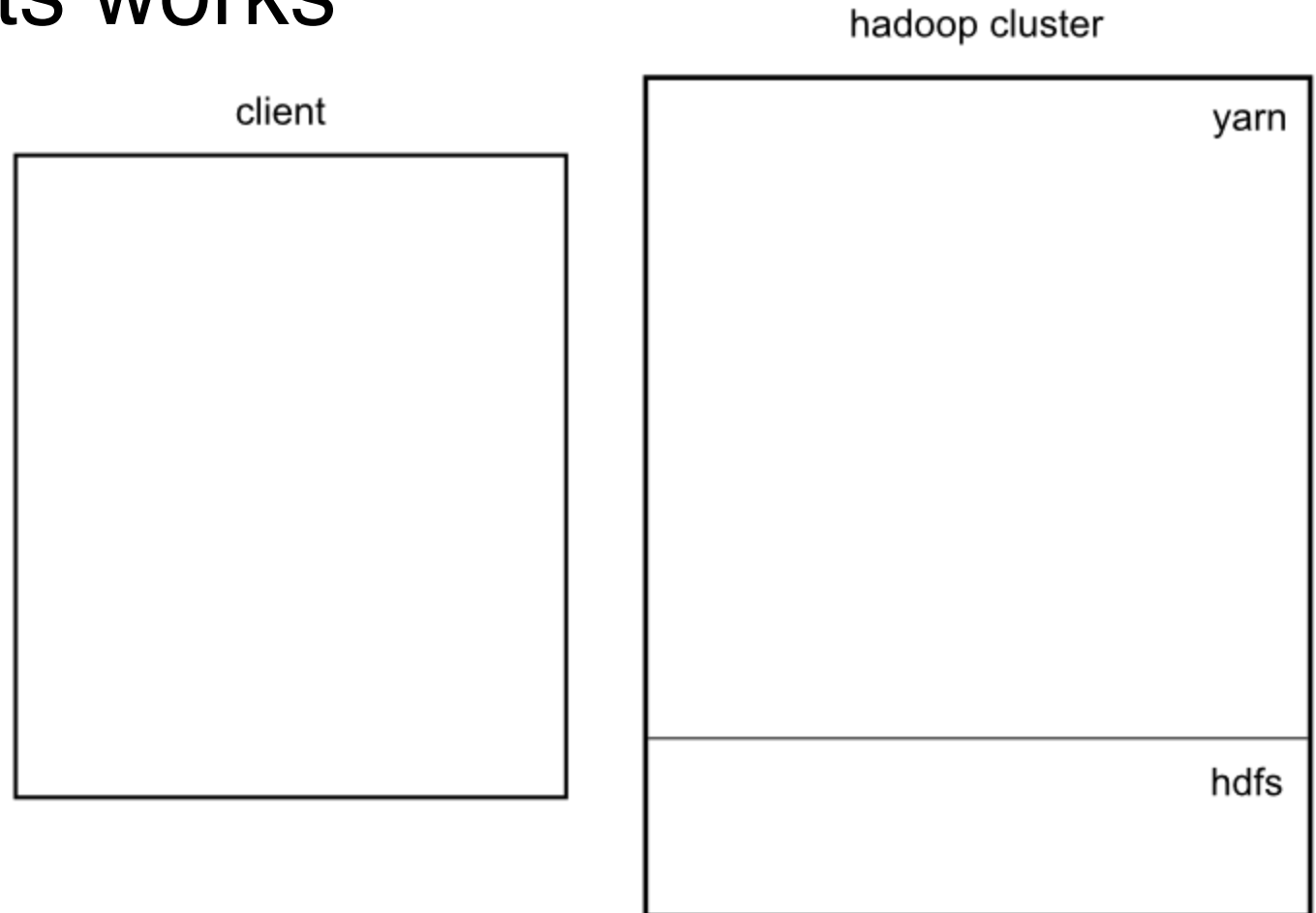


# How does it work

client

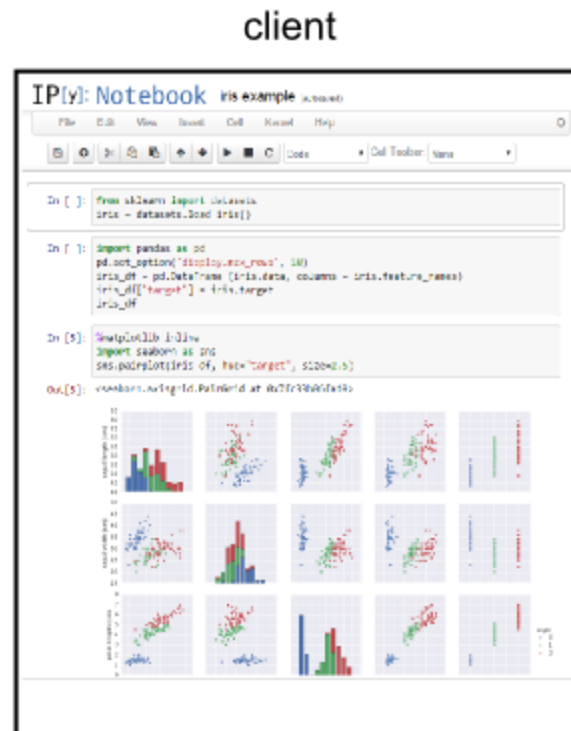


# How does it work

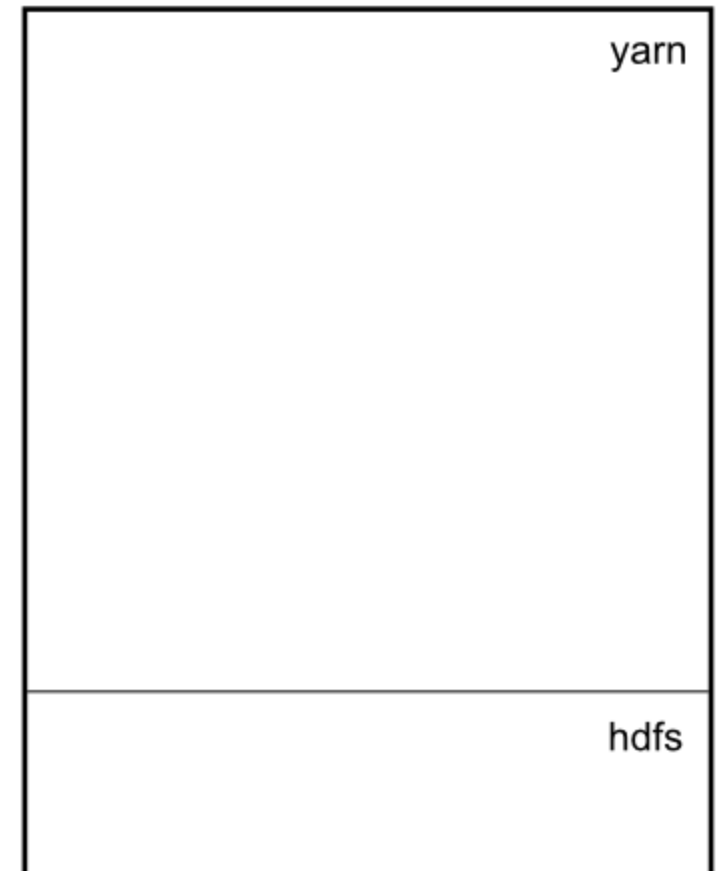




# How does it work

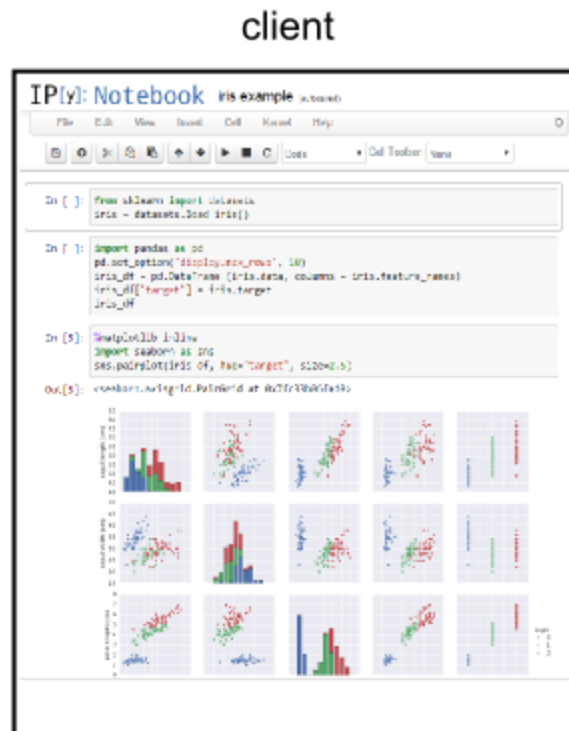


hadoop cluster

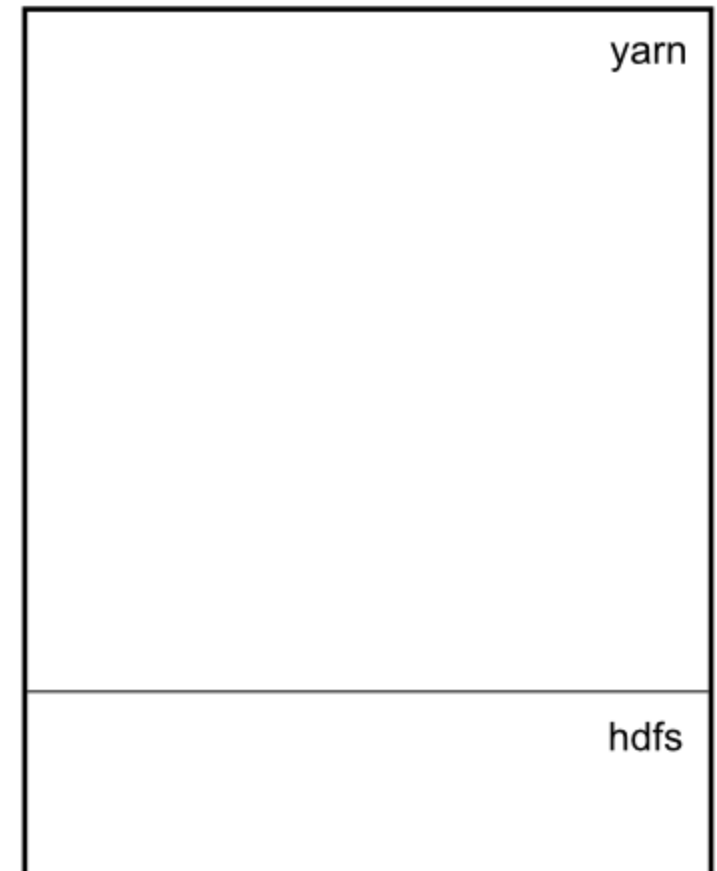


# How does it work

sc = SparkContext( SparkConf() )

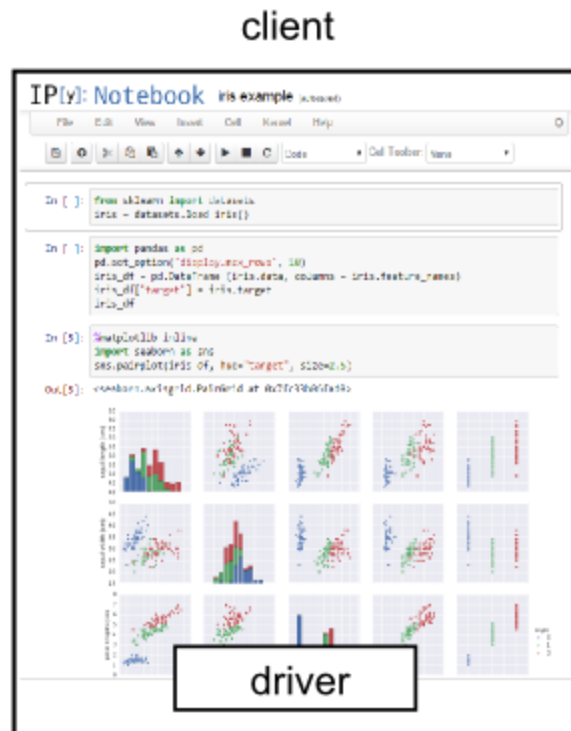


hadoop cluster

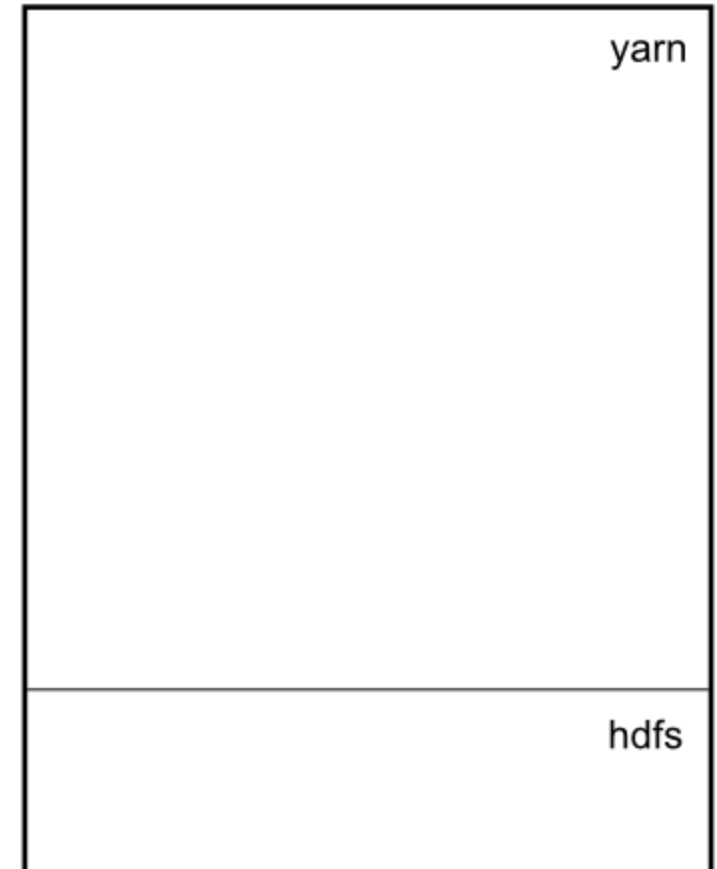


# How does it work

```
sc = SparkContext(SparkConf())
```

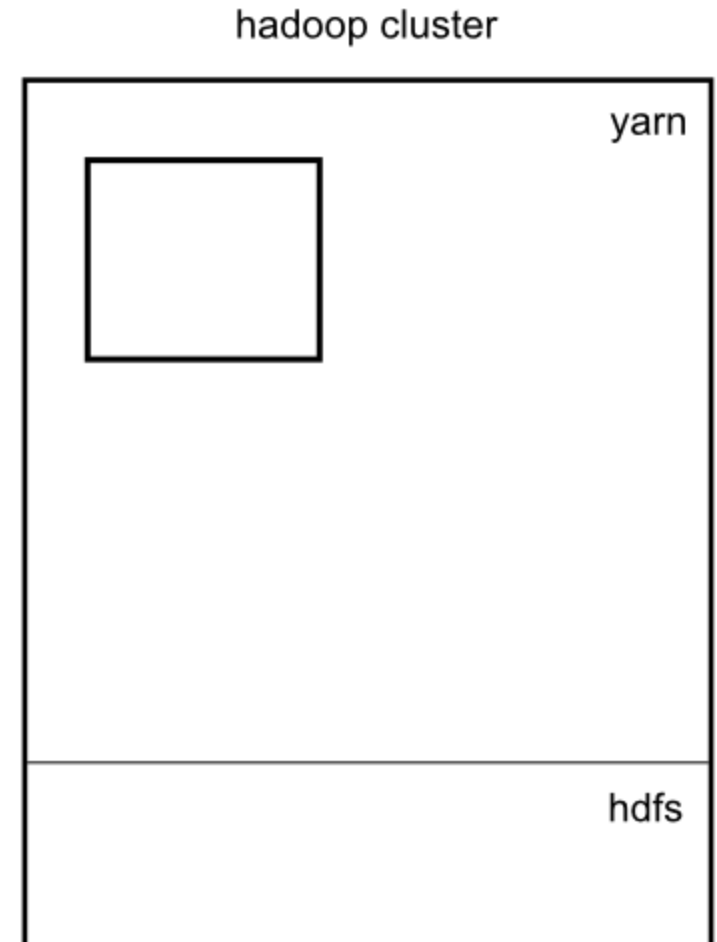
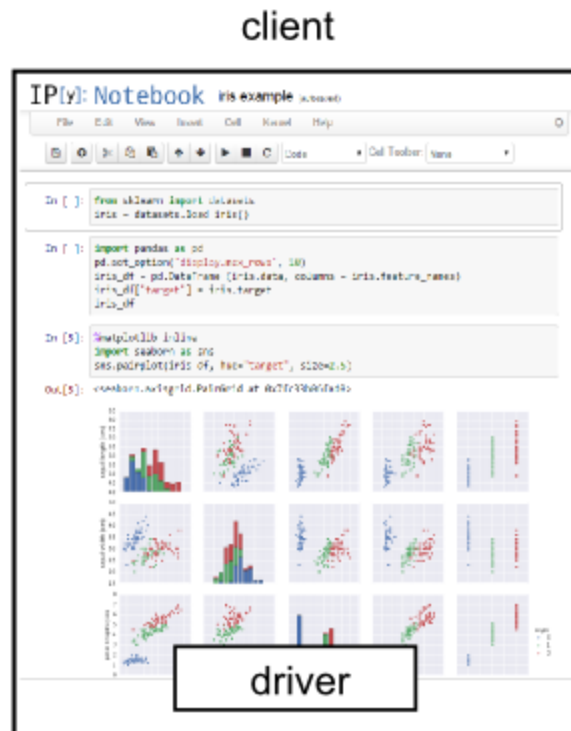


hadoop cluster



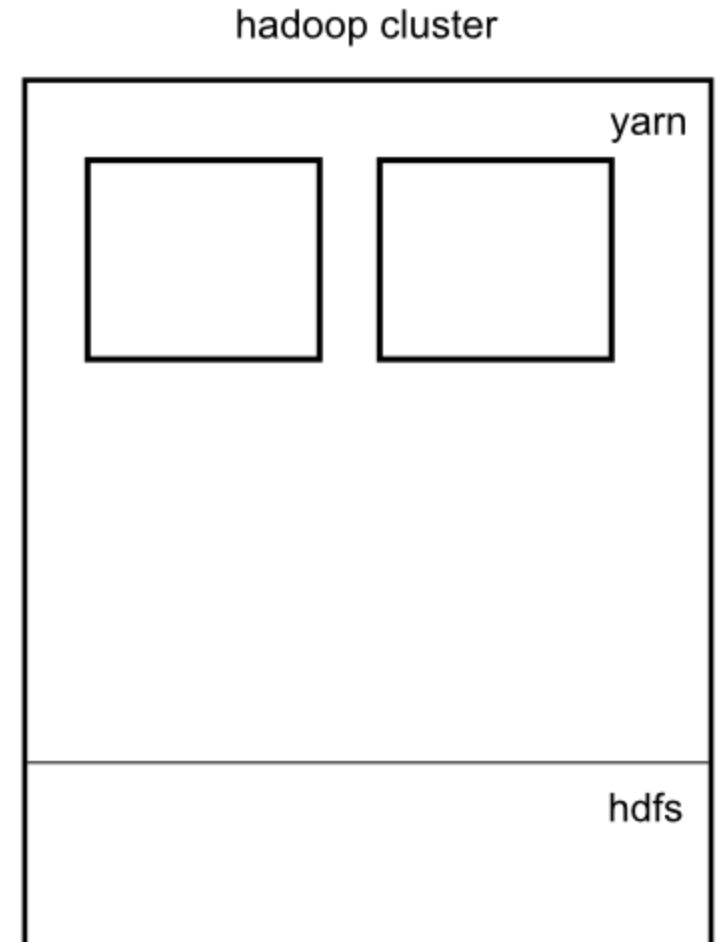
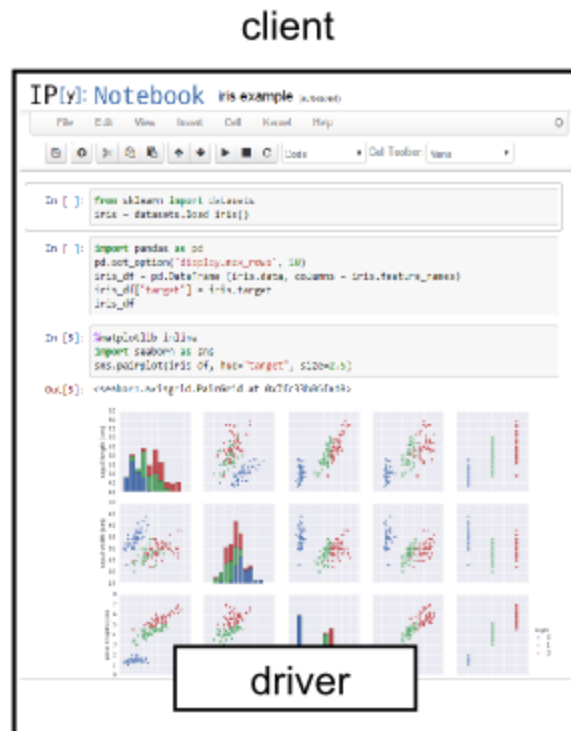
# How does it work

```
sc = SparkContext( SparkConf() )
```



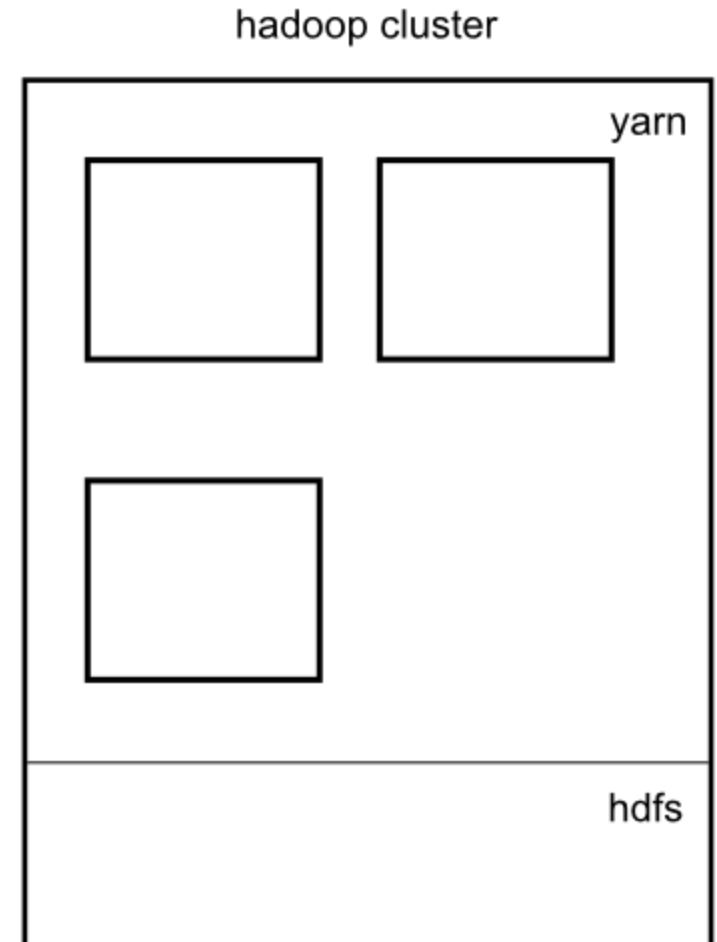
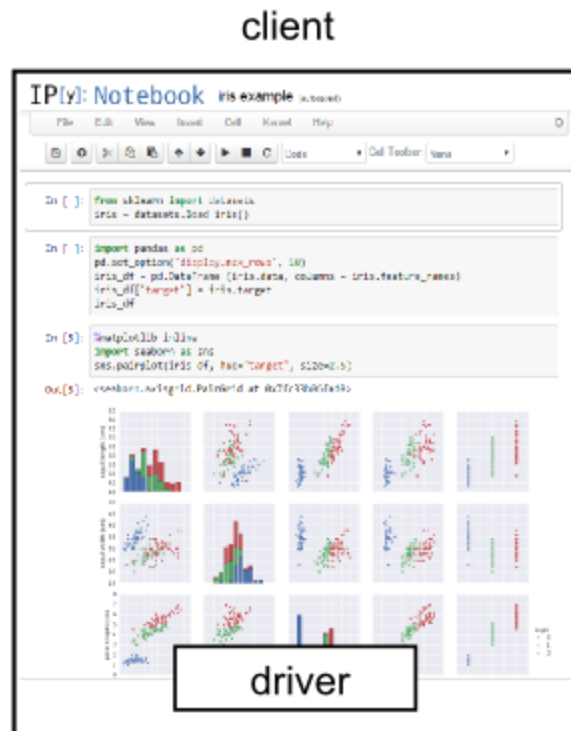
# How does it work

```
sc = SparkContext(SparkConf())
```



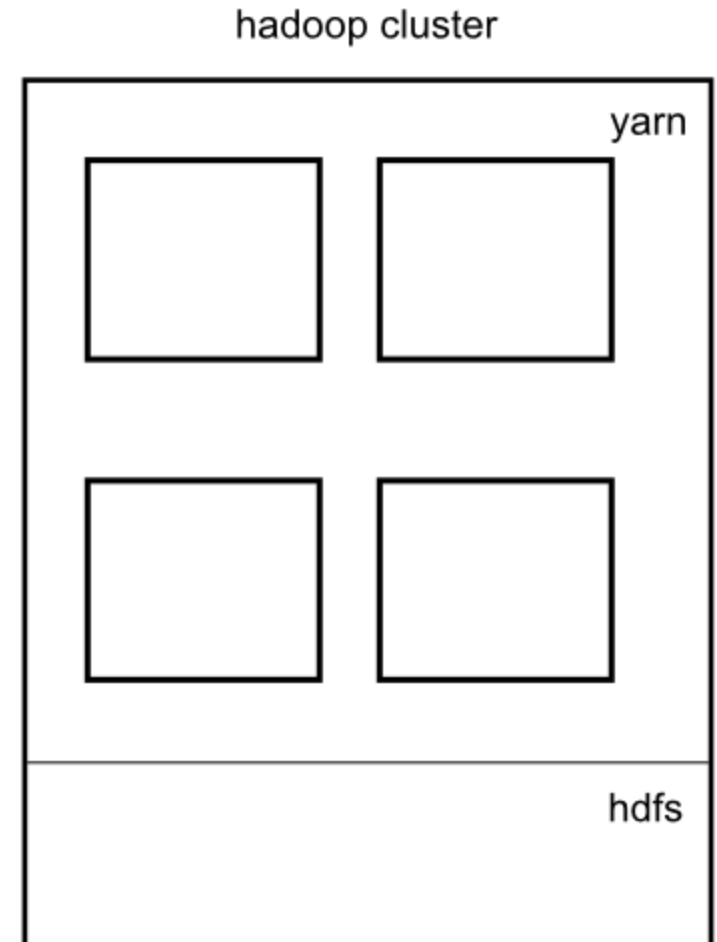
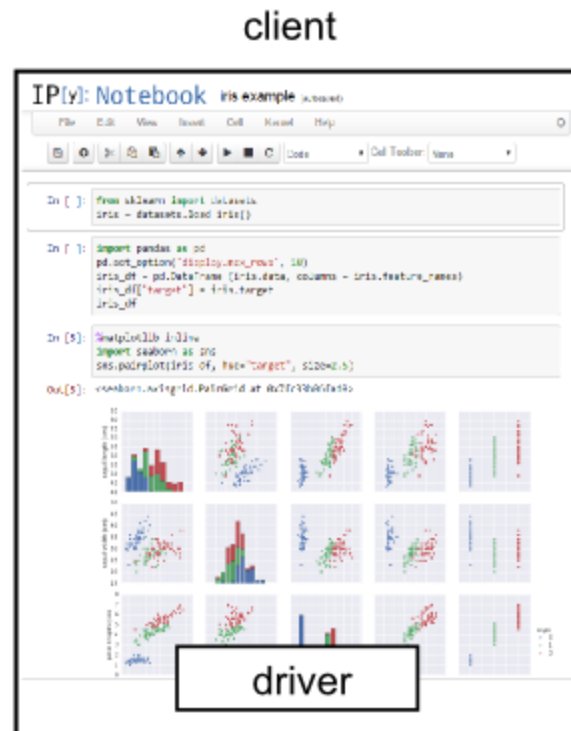
# How does it work

```
sc = SparkContext(SparkConf())
```



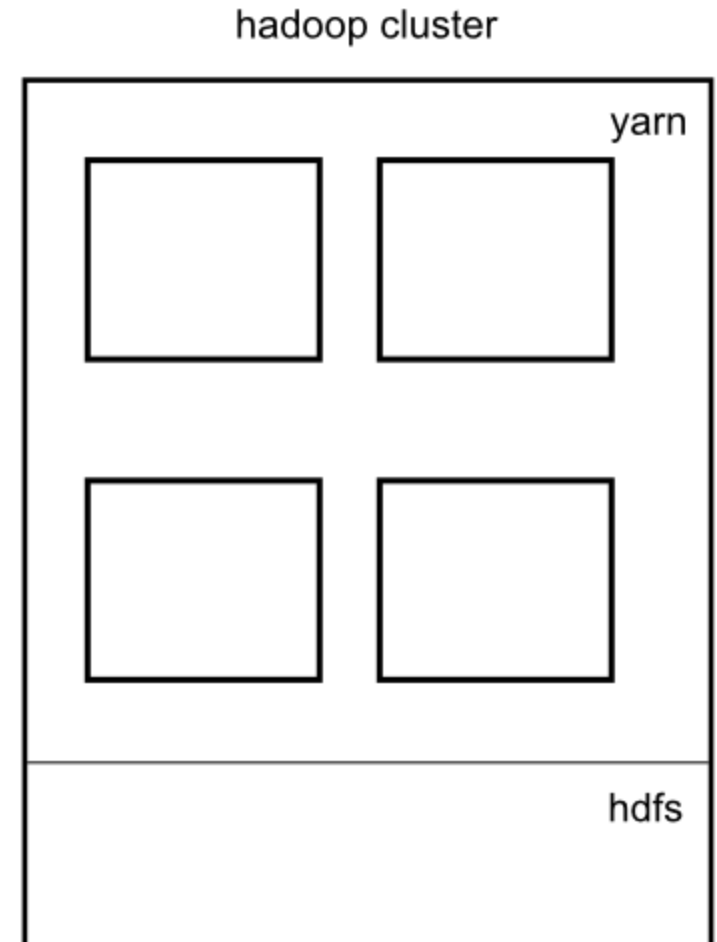
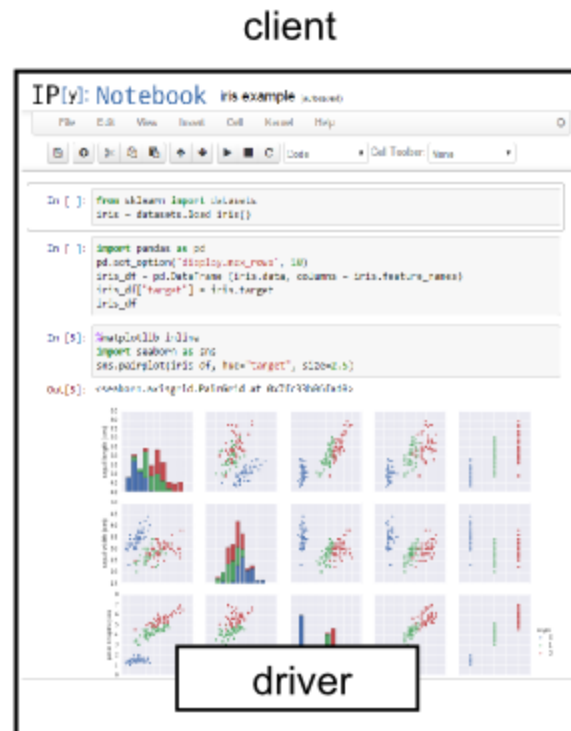
# How does it work

sc = SparkContext( SparkConf() )



# How does it work

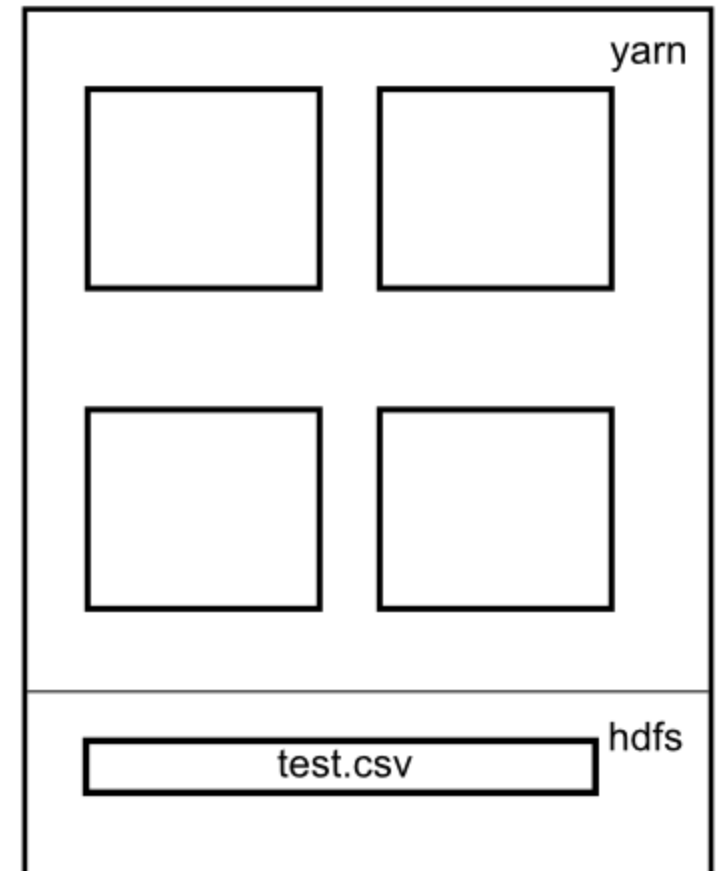
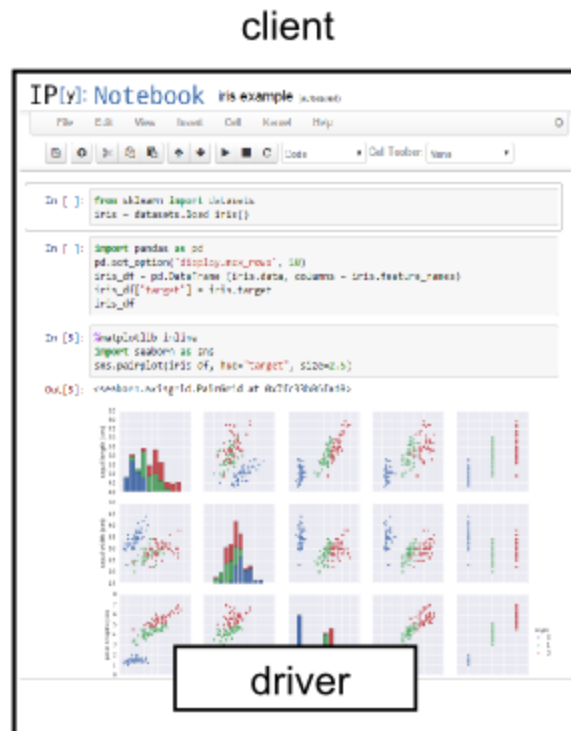
```
a_rdd = sc.textFile("test.csv")
```





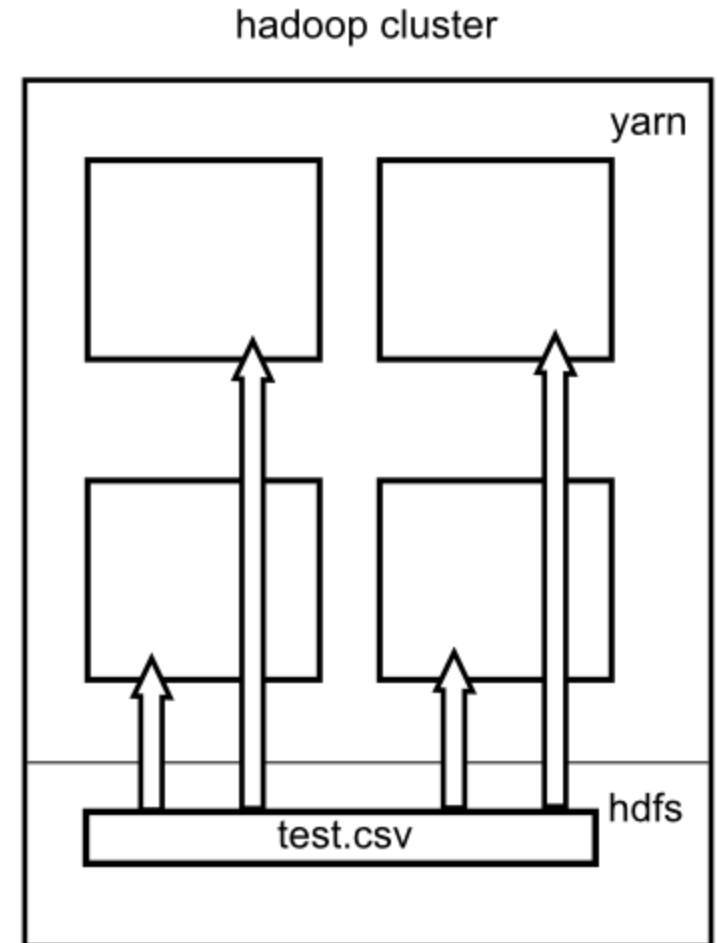
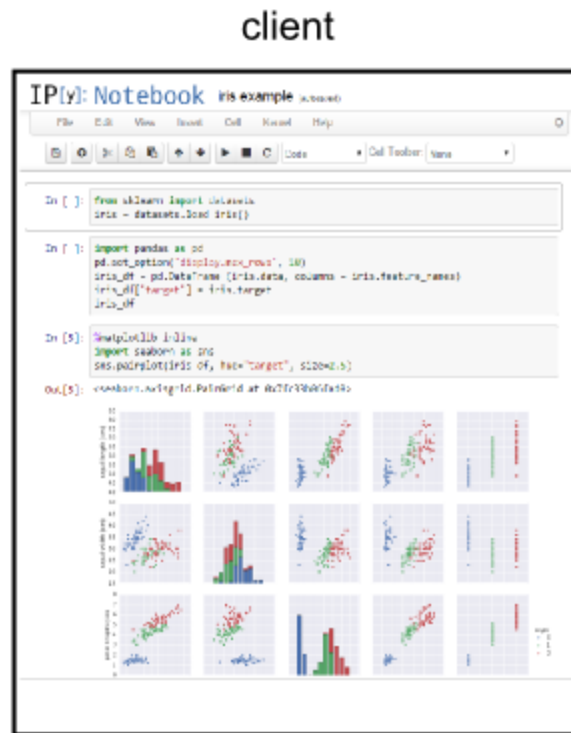
# How does it work

```
a_rdd = sc.textFile("test.csv")
```



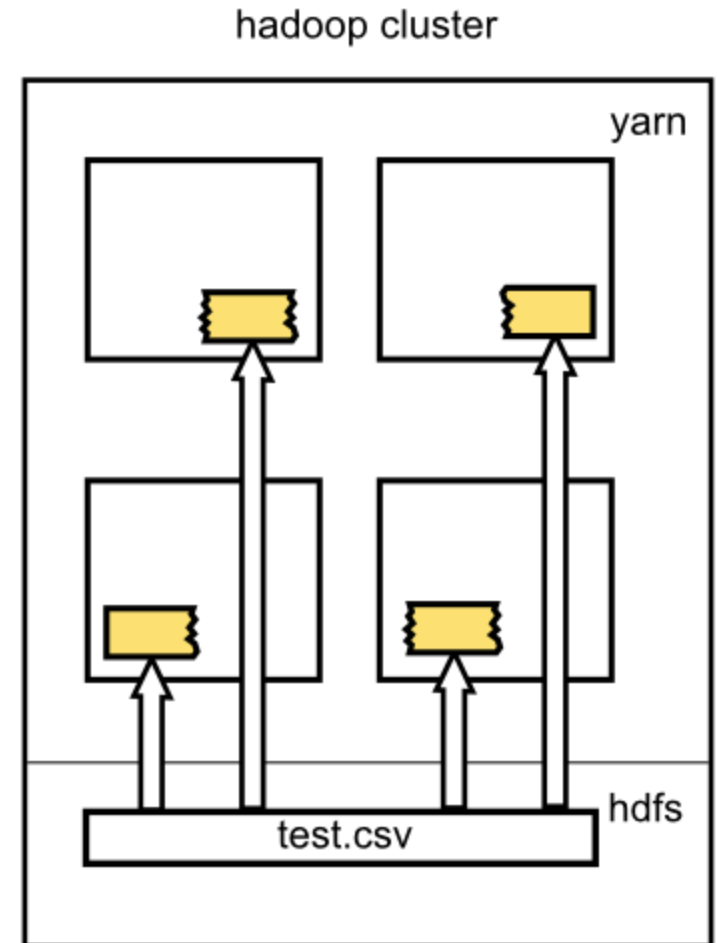
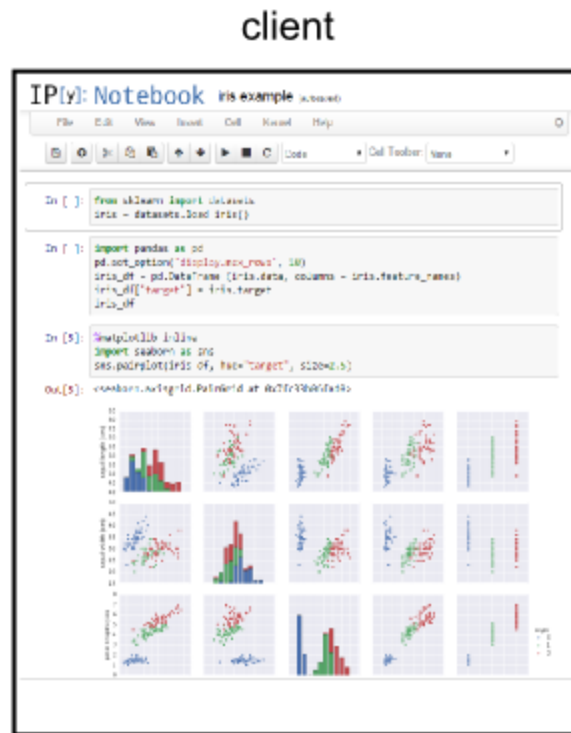
# How does it work

```
a_rdd = sc.textFile("test.csv")
```



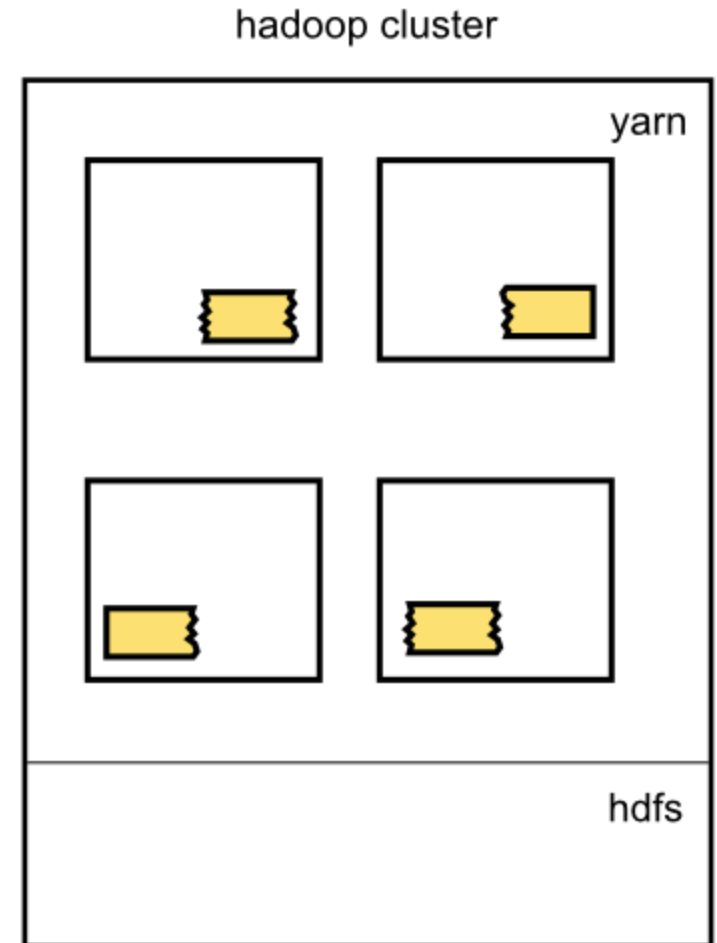
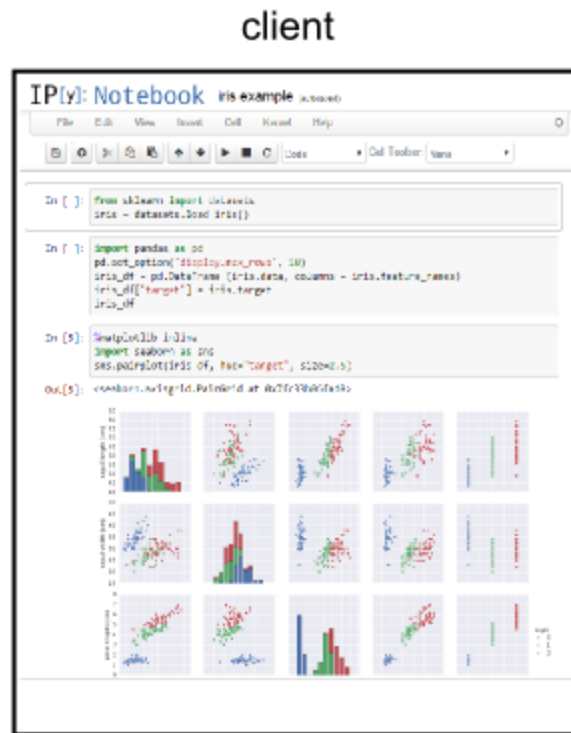
# How does it work

```
a_rdd = sc.textFile("test.csv")
```



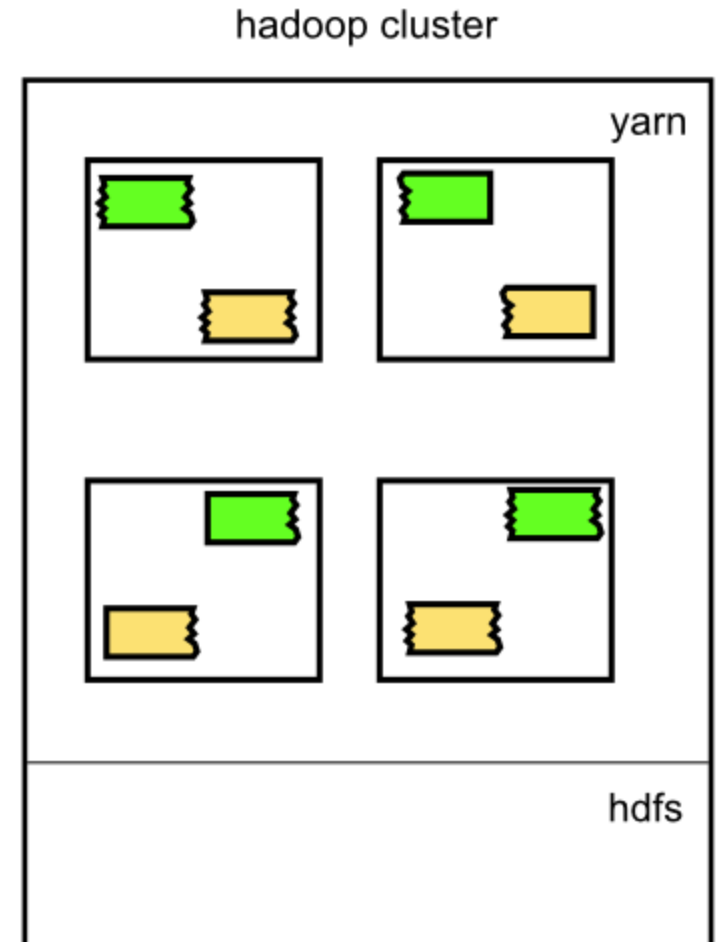
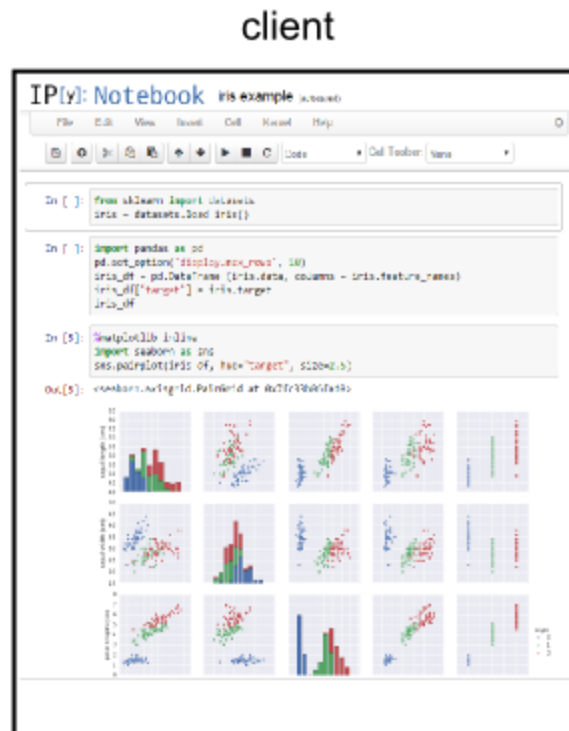
# How does it work

```
b_rdd = a_rdd.map(...)  
c_rdd = b_rdd.reduce(...)  
e_rdd = c_rdd.join(a_rdd)
```



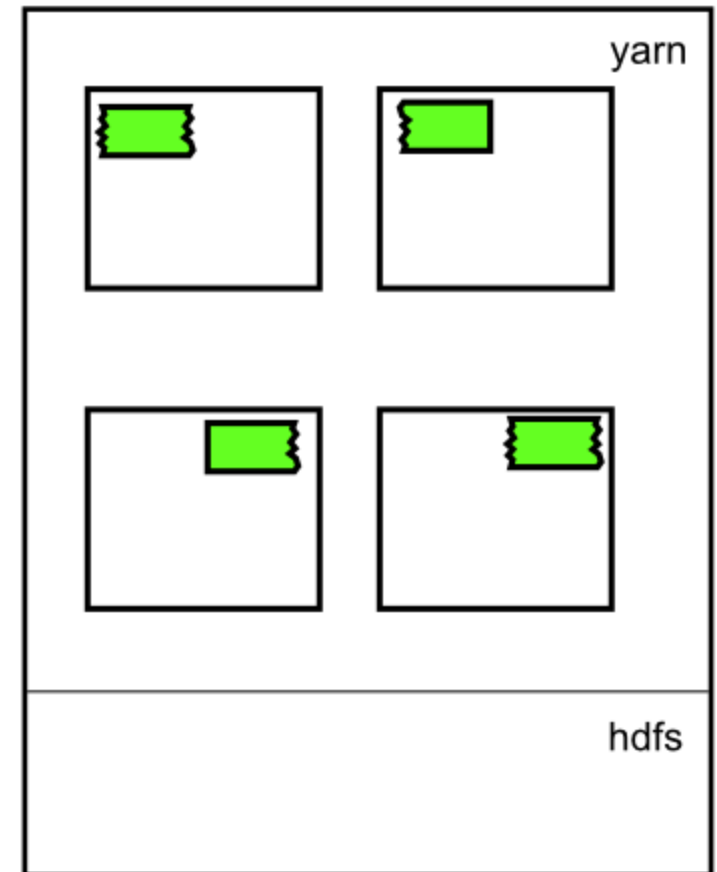
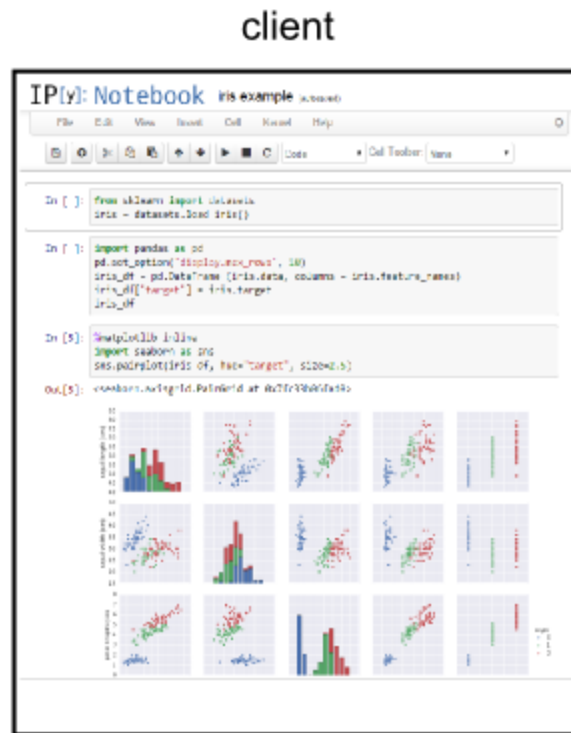
# How does it work

```
b_rdd = a_rdd.map(...)  
c_rdd = b_rdd.reduce(...)  
e_rdd = c_rdd.join(a_rdd)
```



# How does it work

`e_list = e_rdd.collect()`



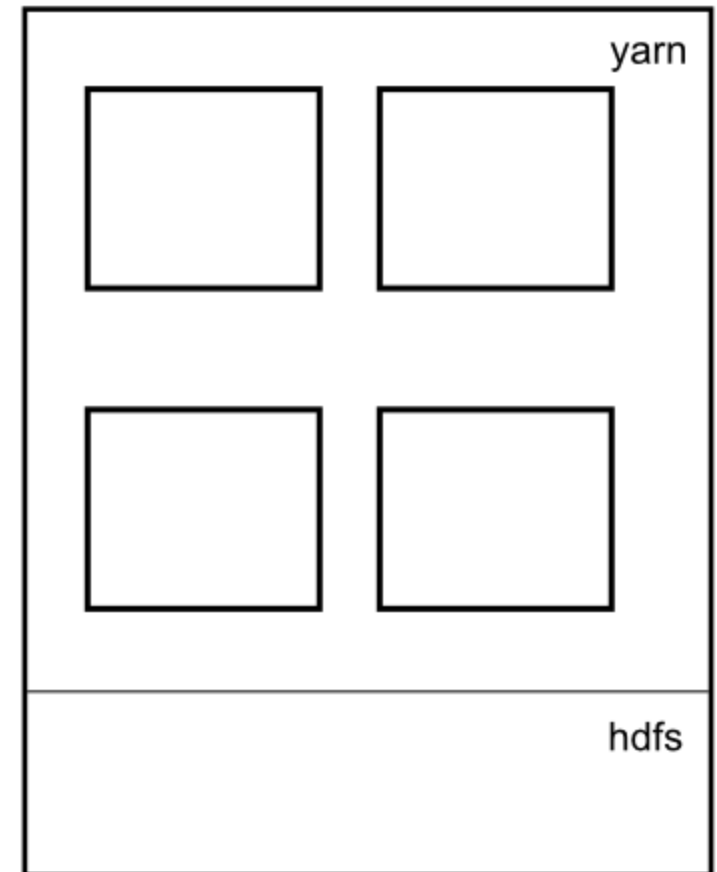
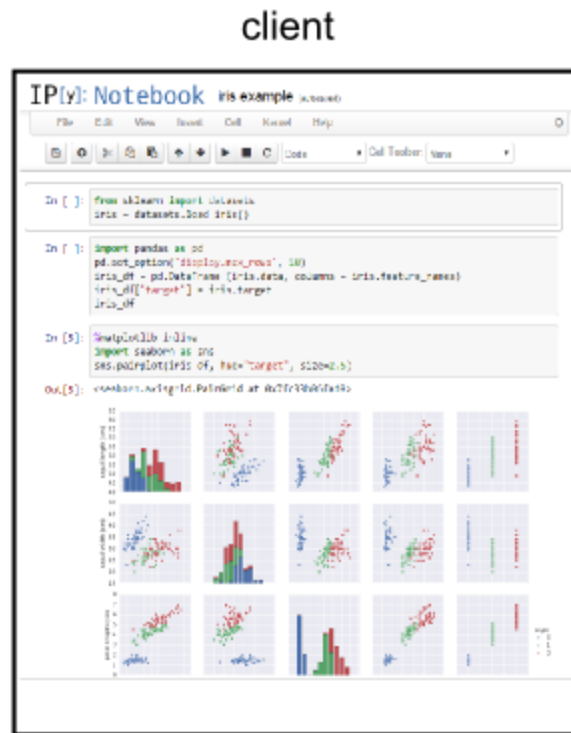
# How does it work

`e_list = e_rdd.collect()`



# How does it work

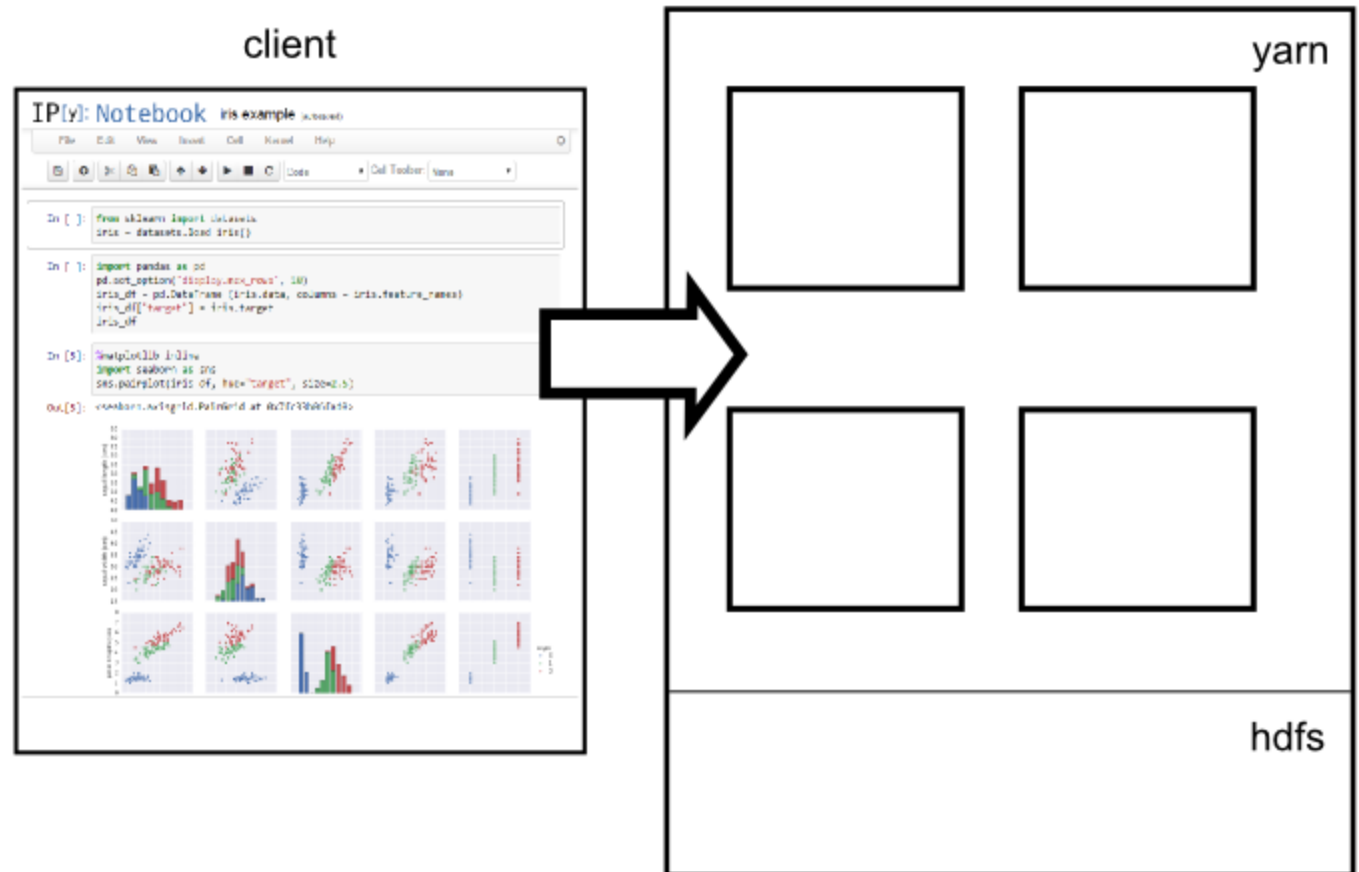
`a_rdd = sc.parallelize(a_list)`





# How does it work

```
a_rdd = sc.parallelize(a_list)
```



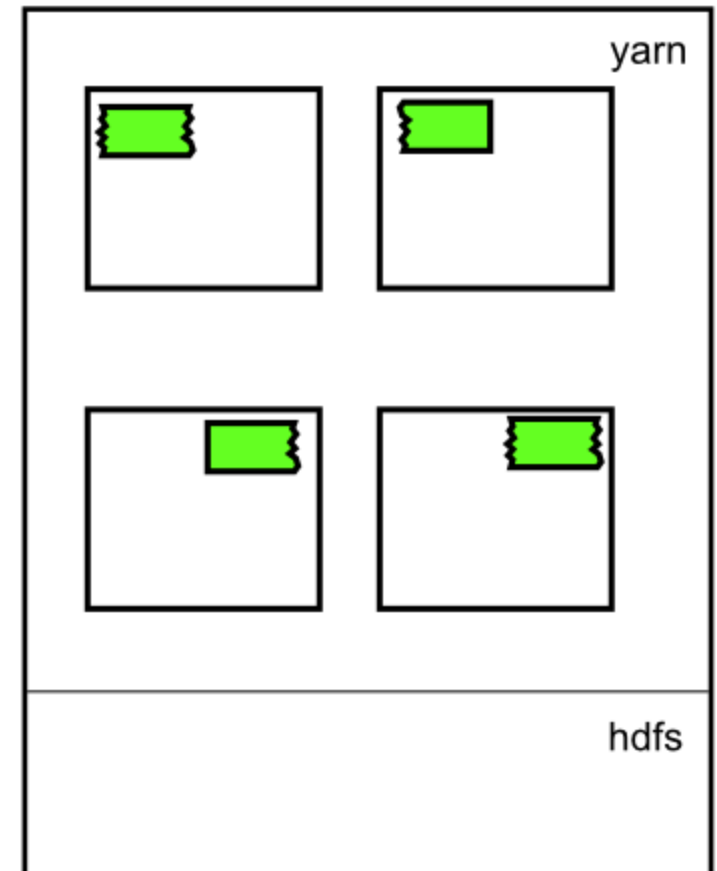
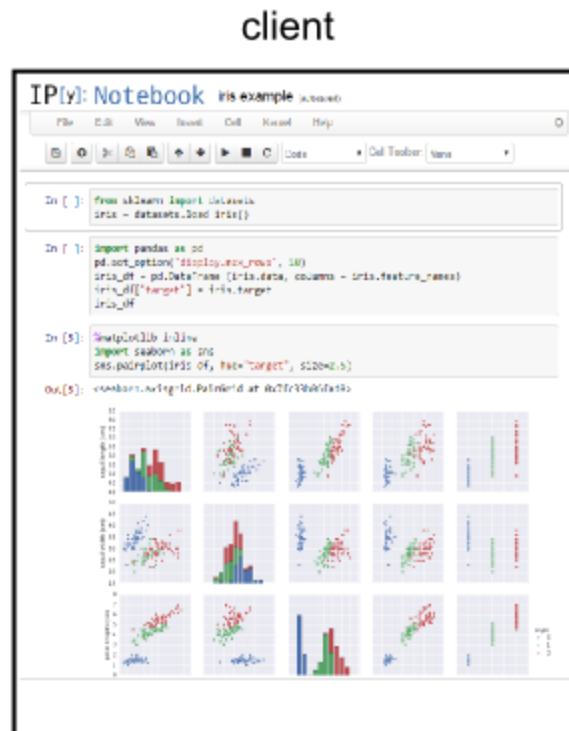
# How does it work

```
a_rdd = sc.parallelize(a_list)
```



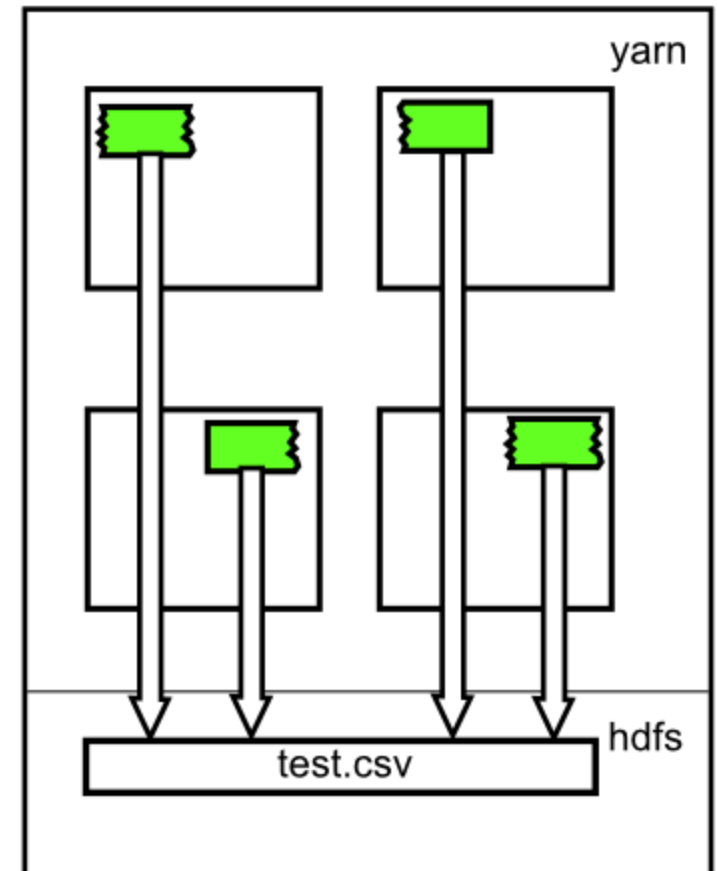
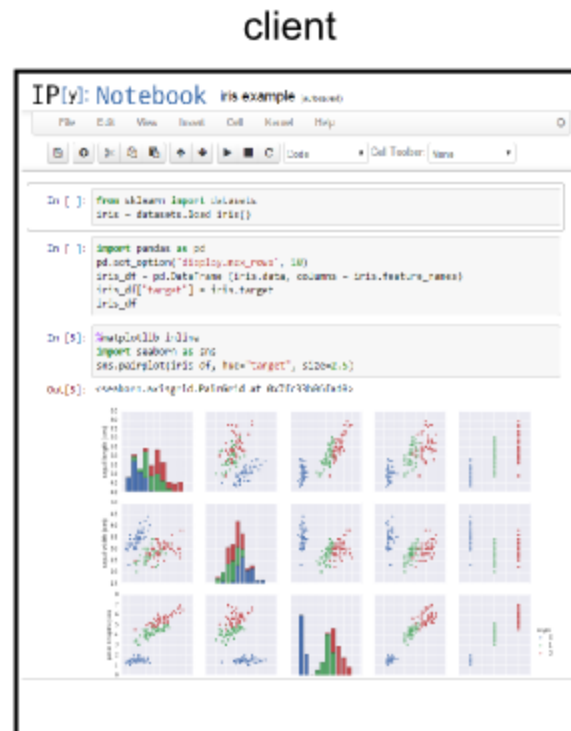
# How does it work

```
a_rdd.saveAsTextFile("test.csv")
```



# How does it work

`a_rdd.saveAsTextFile("test.csv")`



# Apache Spark in Large Scale Machine Learning

# LSML issues

- Too much samples to classify
- Training data does not fit in memory
- Too much training samples
- Too much models to train

# Why not MLlib?

- MLlib is less stable
- too few algorithms comparing to scikit-learn
- ML pipelines are not so mature than in scikit-learn
  - e. g. there is no simple way to use logistic regression for feature selection
- MLlib python API falls behind Java/Scala API
- MLlib is actively developed and may be feasible choice in near future

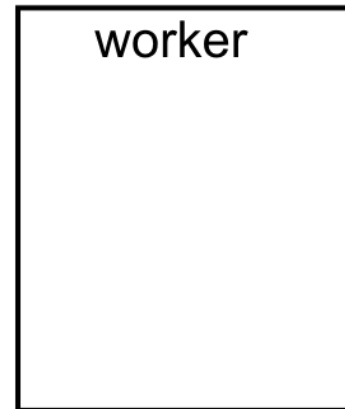
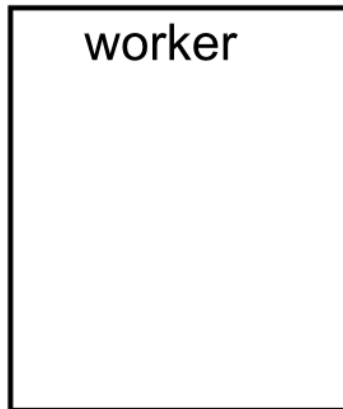
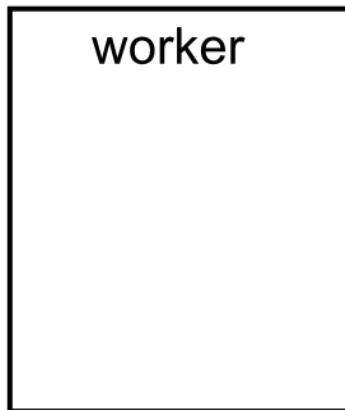
# Spark + scikit-learn = ?

- Parallel training
  - meta-parameter grid search
  - parallel one-vs-rest for multi-class models
  - same features but different targets
  - parallel bagging and ensembles
  - parallel learning for multi-step classification
- Parallel prediction



# Distributed learning

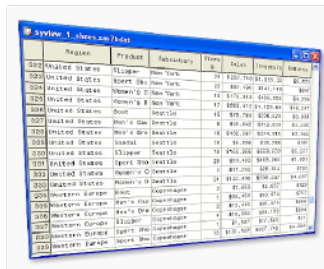
driver



# Distributed learning

dataset

driver



	Product	Category	Price	Weight	Volume
1001	Product 1001	Category 1	100.00	100.00	100.00
1002	Product 1002	Category 2	200.00	200.00	200.00
1003	Product 1003	Category 3	300.00	300.00	300.00
1004	Product 1004	Category 4	400.00	400.00	400.00
1005	Product 1005	Category 5	500.00	500.00	500.00
1006	Product 1006	Category 6	600.00	600.00	600.00
1007	Product 1007	Category 7	700.00	700.00	700.00
1008	Product 1008	Category 8	800.00	800.00	800.00
1009	Product 1009	Category 9	900.00	900.00	900.00
1010	Product 1010	Category 10	1000.00	1000.00	1000.00

worker

worker

worker

# Distributed learning

dataset

[illegible]

driver

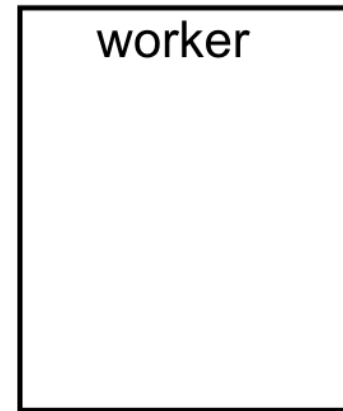
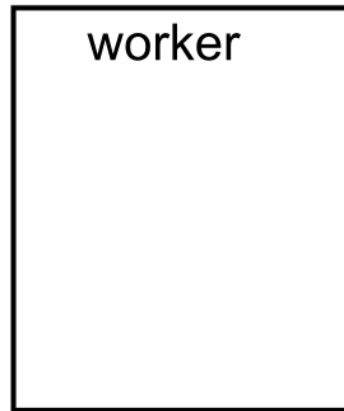
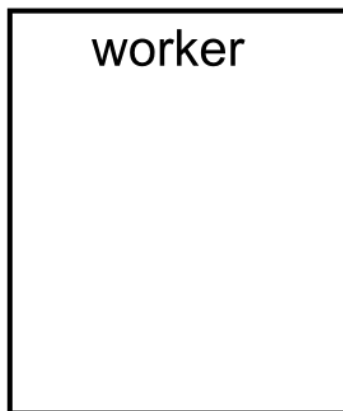
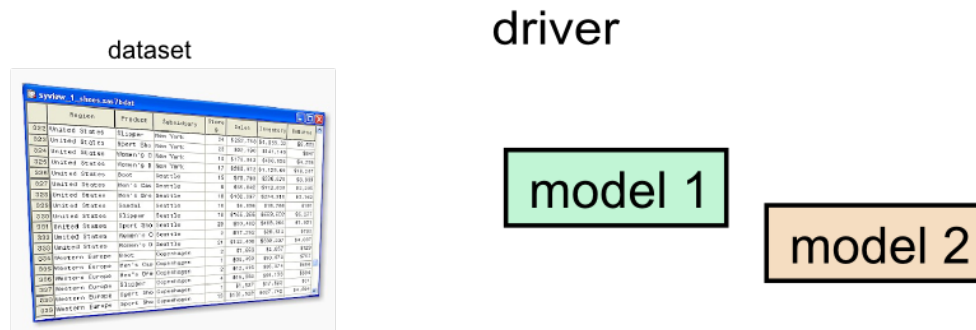
model 1

worker

worker

worker

# Distributed learning



# Distributed learning

dataset

driver

Appendix 1: Sales Analysis							
	Region	Product	Subcategory	Item #	Quantity	Unit Price	Unit Cost
323	United States	Blimpie	Hot Veggies	24	100	\$2.00	\$1.50
324	United States	Blondie	Hot Veggies	25	100	\$2.00	\$1.50
325	United States	Blondie	Hot Veggies	26	100	\$2.00	\$1.50
326	United States	Blondie	Hot Veggies	27	100	\$2.00	\$1.50
327	United States	Blondie	Hot Veggies	28	100	\$2.00	\$1.50
328	United States	Blondie	Hot Veggies	29	100	\$2.00	\$1.50
329	United States	Blondie	Hot Veggies	30	100	\$2.00	\$1.50
330	United States	Blondie	Hot Veggies	31	100	\$2.00	\$1.50
331	United States	Blondie	Hot Veggies	32	100	\$2.00	\$1.50
332	United States	Blondie	Hot Veggies	33	100	\$2.00	\$1.50
333	United States	Blondie	Hot Veggies	34	100	\$2.00	\$1.50
334	United States	Blondie	Hot Veggies	35	100	\$2.00	\$1.50
335	United States	Blondie	Hot Veggies	36	100	\$2.00	\$1.50
336	United States	Blondie	Hot Veggies	37	100	\$2.00	\$1.50
337	United States	Blondie	Hot Veggies	38	100	\$2.00	\$1.50
338	United States	Blondie	Hot Veggies	39	100	\$2.00	\$1.50
339	United States	Blondie	Hot Veggies	40	100	\$2.00	\$1.50
340	United States	Blondie	Hot Veggies	41	100	\$2.00	\$1.50
341	United States	Blondie	Hot Veggies	42	100	\$2.00	\$1.50
342	United States	Blondie	Hot Veggies	43	100	\$2.00	\$1.50
343	United States	Blondie	Hot Veggies	44	100	\$2.00	\$1.50
344	United States	Blondie	Hot Veggies	45	100	\$2.00	\$1.50
345	United States	Blondie	Hot Veggies	46	100	\$2.00	\$1.50
346	United States	Blondie	Hot Veggies	47	100	\$2.00	\$1.50
347	United States	Blondie	Hot Veggies	48	100	\$2.00	\$1.50
348	United States	Blondie	Hot Veggies	49	100	\$2.00	\$1.50
349	United States	Blondie	Hot Veggies	50	100	\$2.00	\$1.50
350	United States	Blondie	Hot Veggies	51	100	\$2.00	\$1.50
351	United States	Blondie	Hot Veggies	52	100	\$2.00	\$1.50
352	United States	Blondie	Hot Veggies	53	100	\$2.00	\$1.50
353	United States	Blondie	Hot Veggies	54	100	\$2.00	\$1.50
354	United States	Blondie	Hot Veggies	55	100	\$2.00	\$1.50
355	United States	Blondie	Hot Veggies	56	100	\$2.00	\$1.50
356	United States	Blondie	Hot Veggies	57	100	\$2.00	\$1.50
357	United States	Blondie	Hot Veggies	58	100	\$2.00	\$1.50
358	United States	Blondie	Hot Veggies	59	100	\$2.00	\$1.50
359	United States	Blondie	Hot Veggies	60	100	\$2.00	\$1.50
360	United States	Blondie	Hot Veggies	61	100	\$2.00	\$1.50
361	United States	Blondie	Hot Veggies	62	100	\$2.00	\$1.50
362	United States	Blondie	Hot Veggies	63	100	\$2.00	\$1.50
363	United States	Blondie	Hot Veggies	64	100	\$2.00	\$1.50
364	United States	Blondie	Hot Veggies	65	100	\$2.00	\$1.50
365	United States	Blondie	Hot Veggies	66	100	\$2.00	\$1.50
366	United States	Blondie	Hot Veggies	67	100	\$2.00	\$1.50
367	United States	Blondie	Hot Veggies	68	100	\$2.00	\$1.50
368	United States	Blondie	Hot Veggies	69	100	\$2.00	\$1.50
369	United States	Blondie	Hot Veggies	70	100	\$2.00	\$1.50
370	United States	Blondie	Hot Veggies	71	100	\$2.00	\$1.50
371	United States	Blondie	Hot Veggies	72	100	\$2.00	\$1.50
372	United States	Blondie	Hot Veggies	73	100	\$2.00	\$1.50
373	United States	Blondie	Hot Veggies	74	100	\$2.00	\$1.50

model 1

model 2

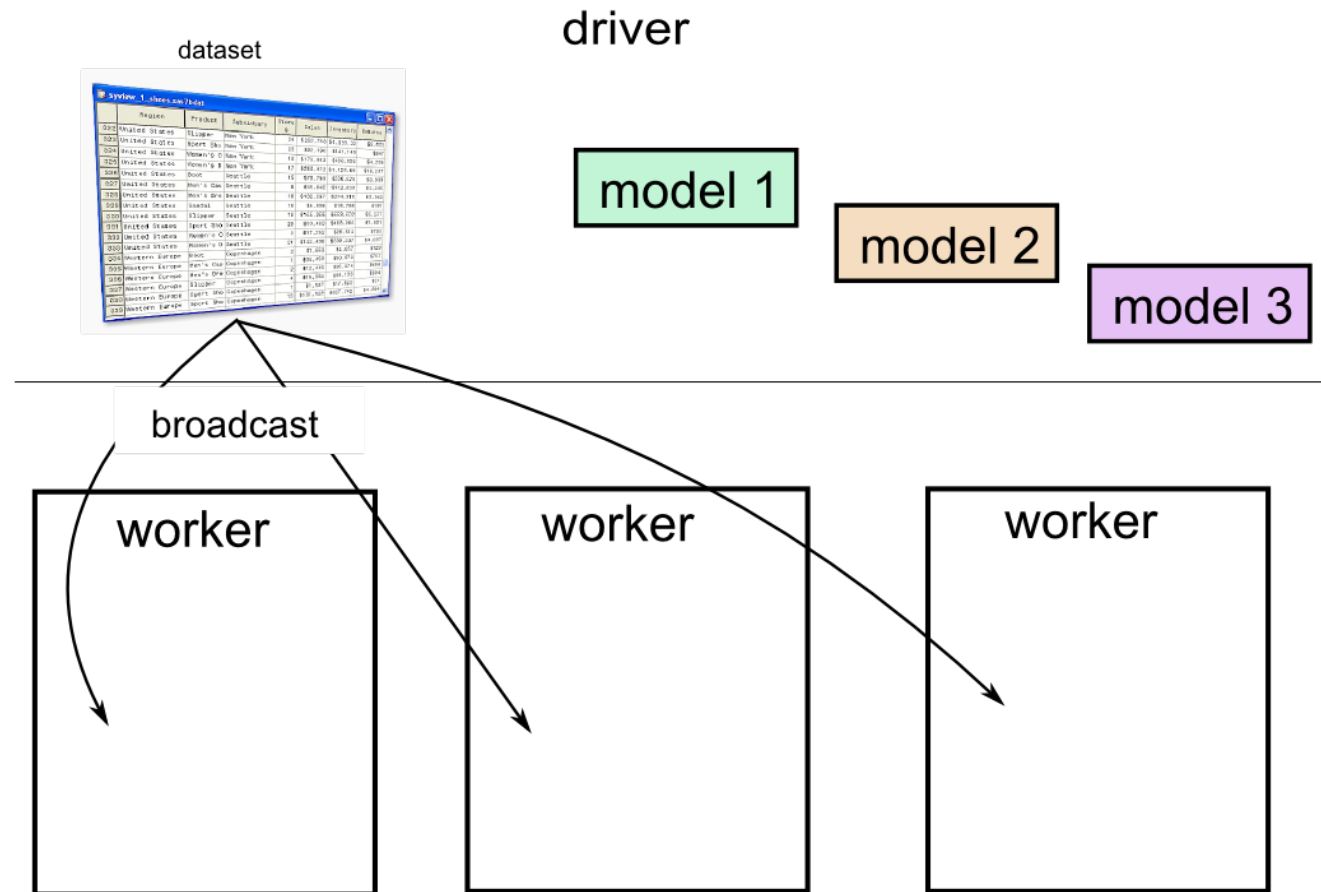
model 3

worker

worker

worker

# Distributed learning



# Distributed learning

dataset

driver

model 1

model 2

### model 3

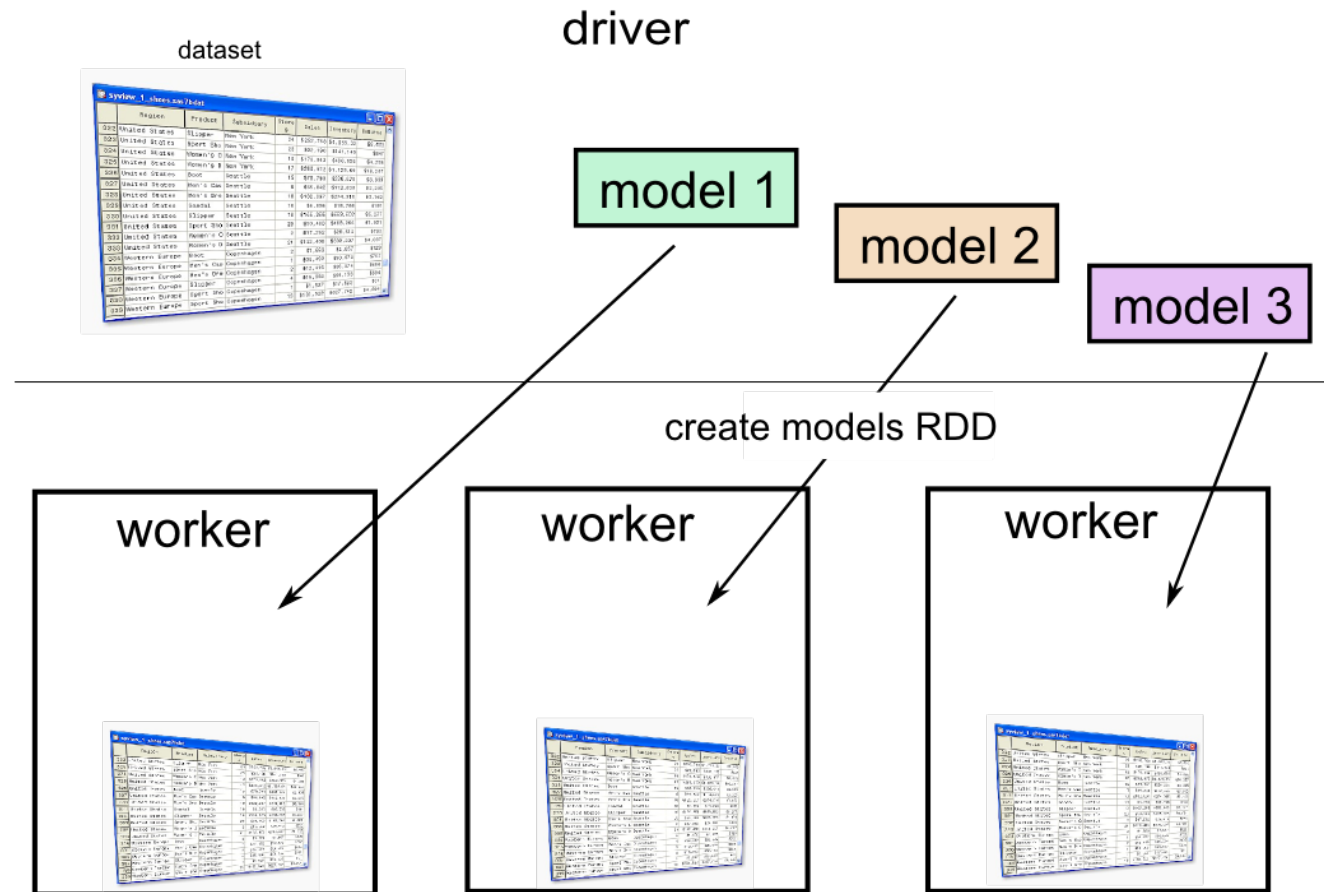
broadcast

worker

worker

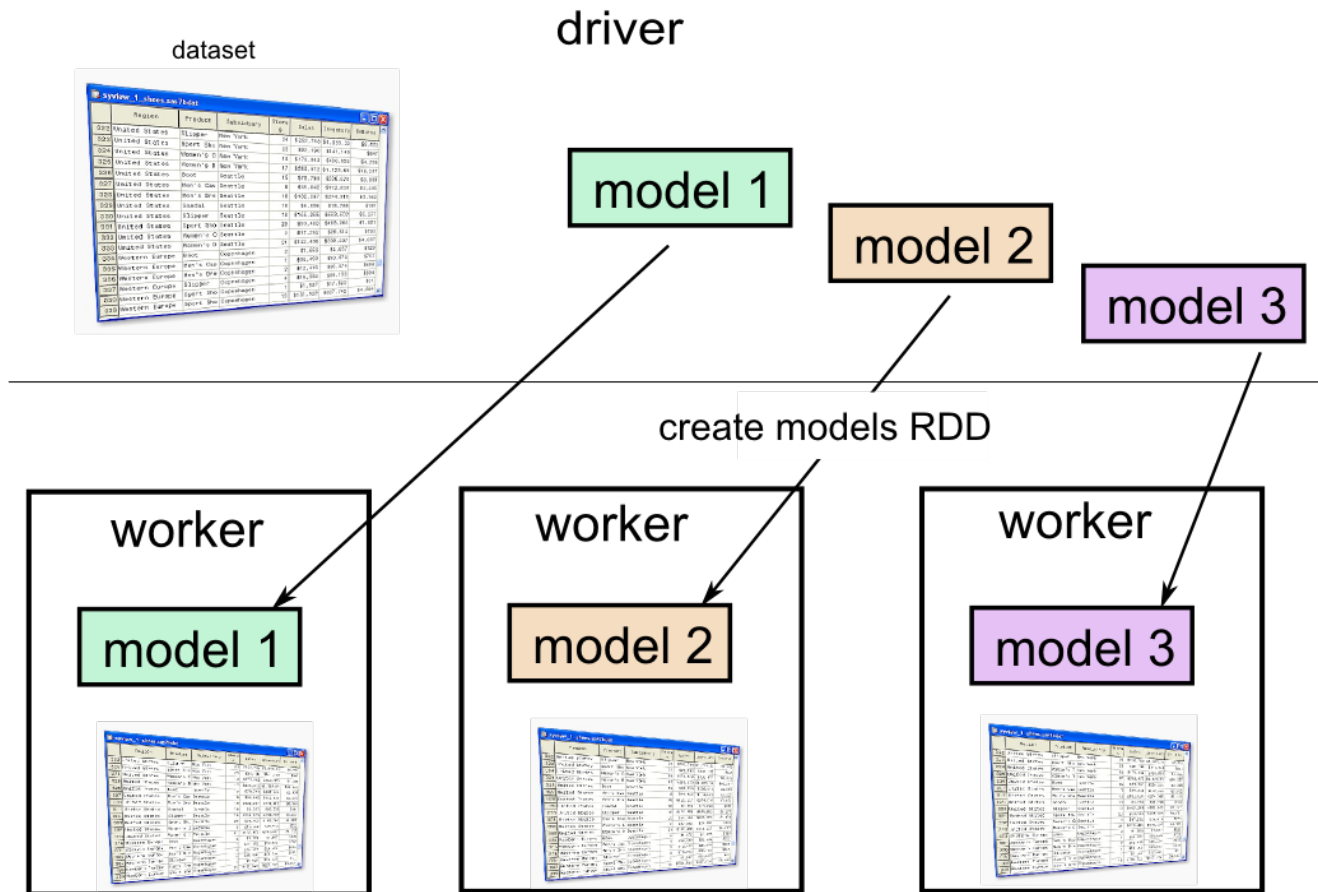
worker

# Distributed learning



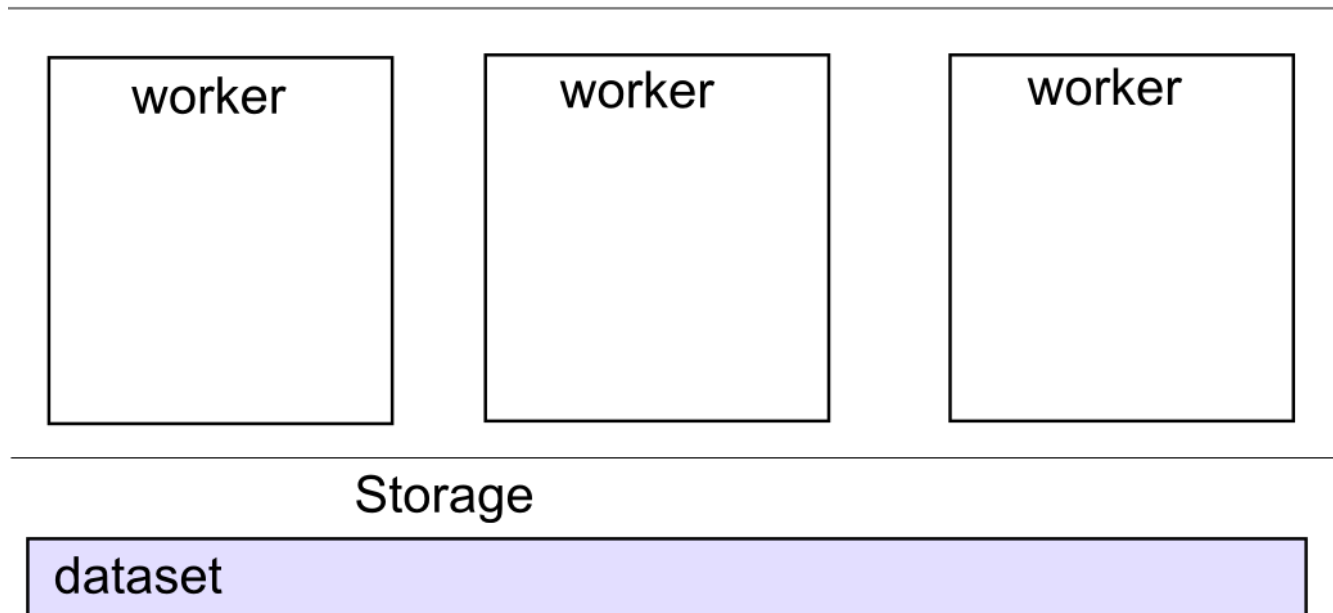


# Distributed learning



# Distributed prediction

driver



# Distributed prediction

driver

model 1

---

worker

worker

worker

---

Storage

dataset

# Distributed prediction

driver

model 1

model 2

---

worker

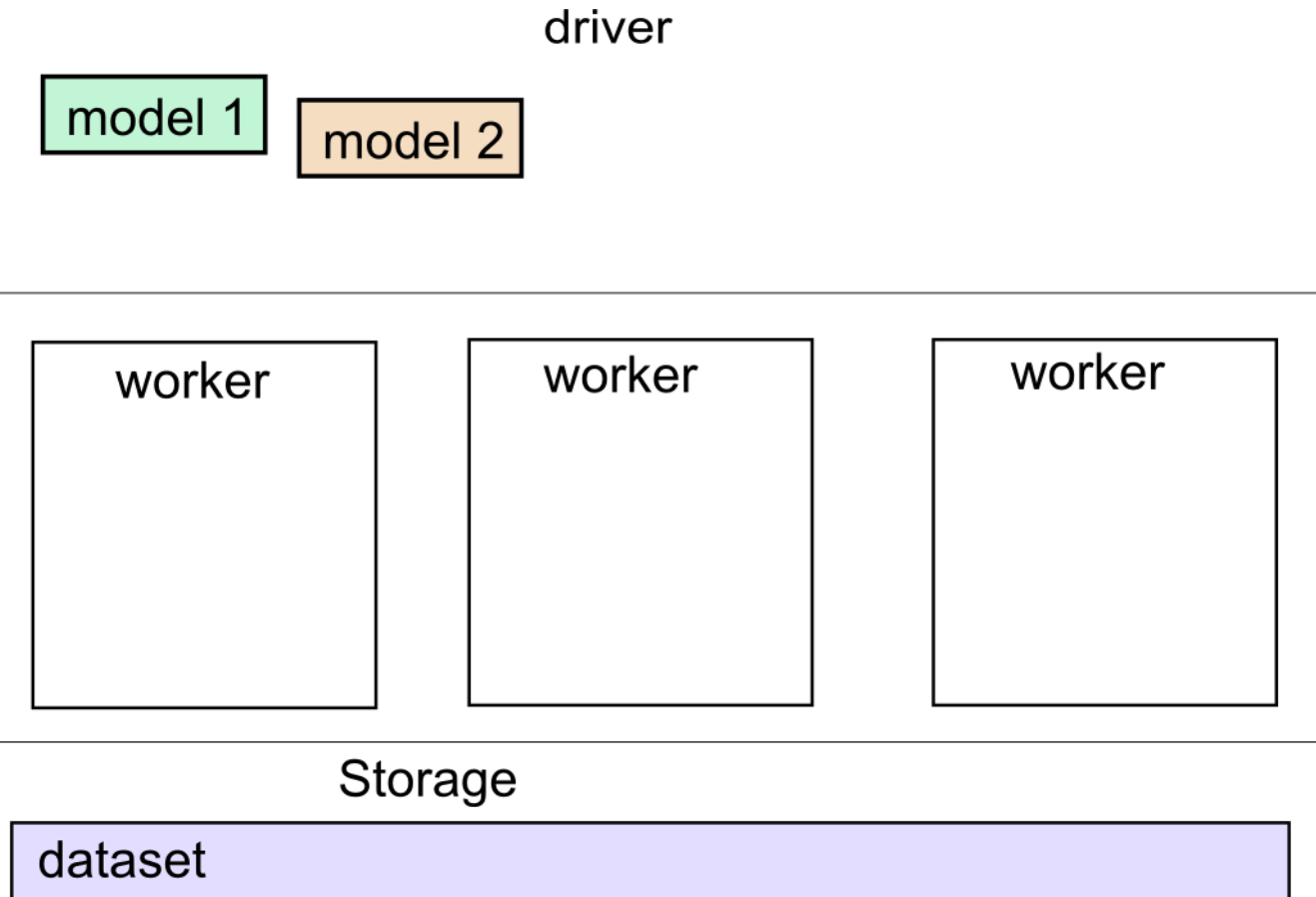
worker

worker

---

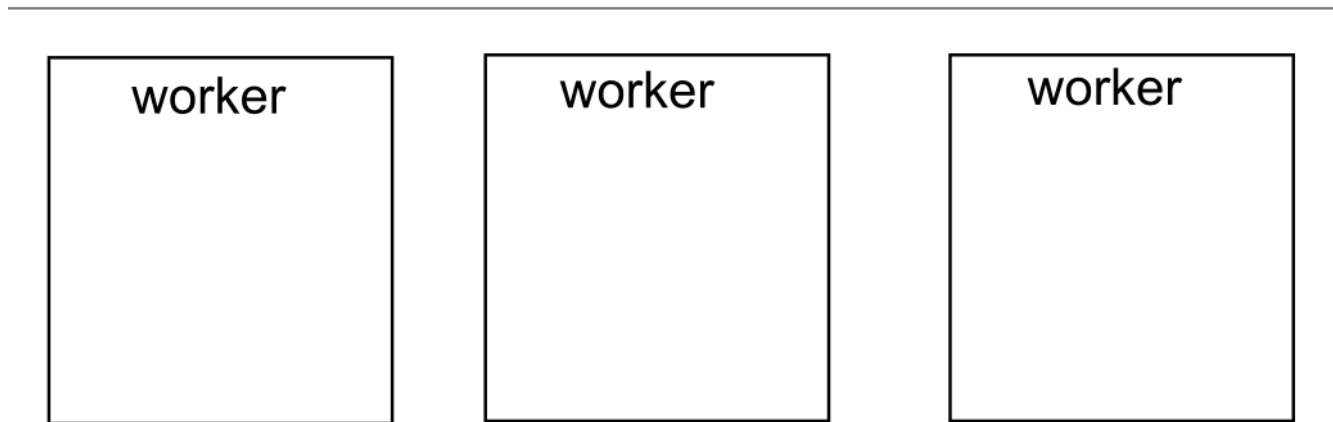
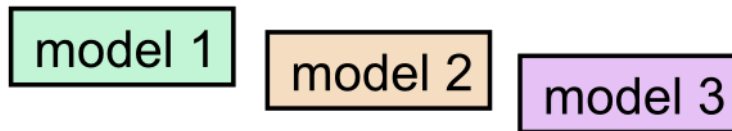
Storage

dataset



# Distributed prediction

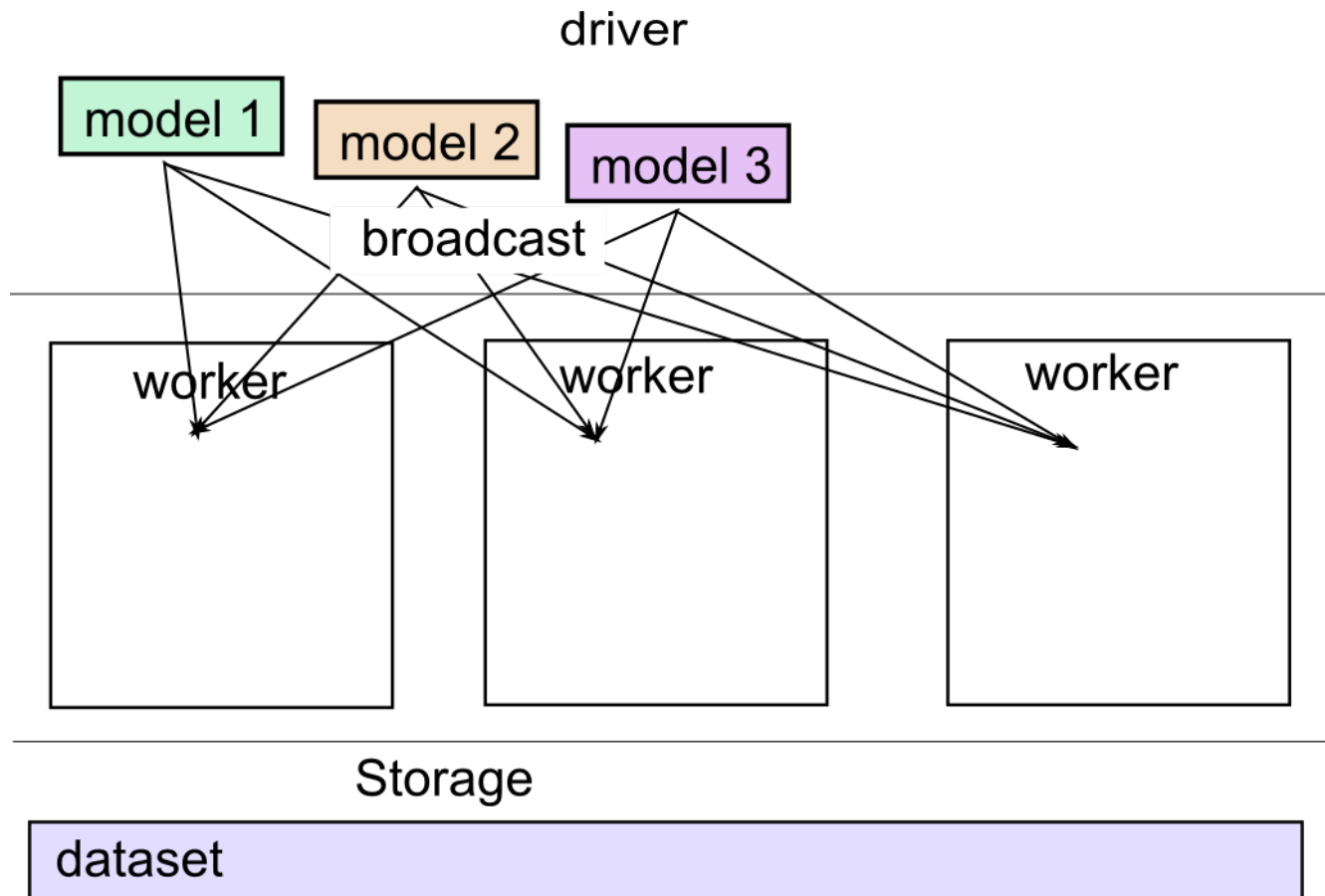
driver



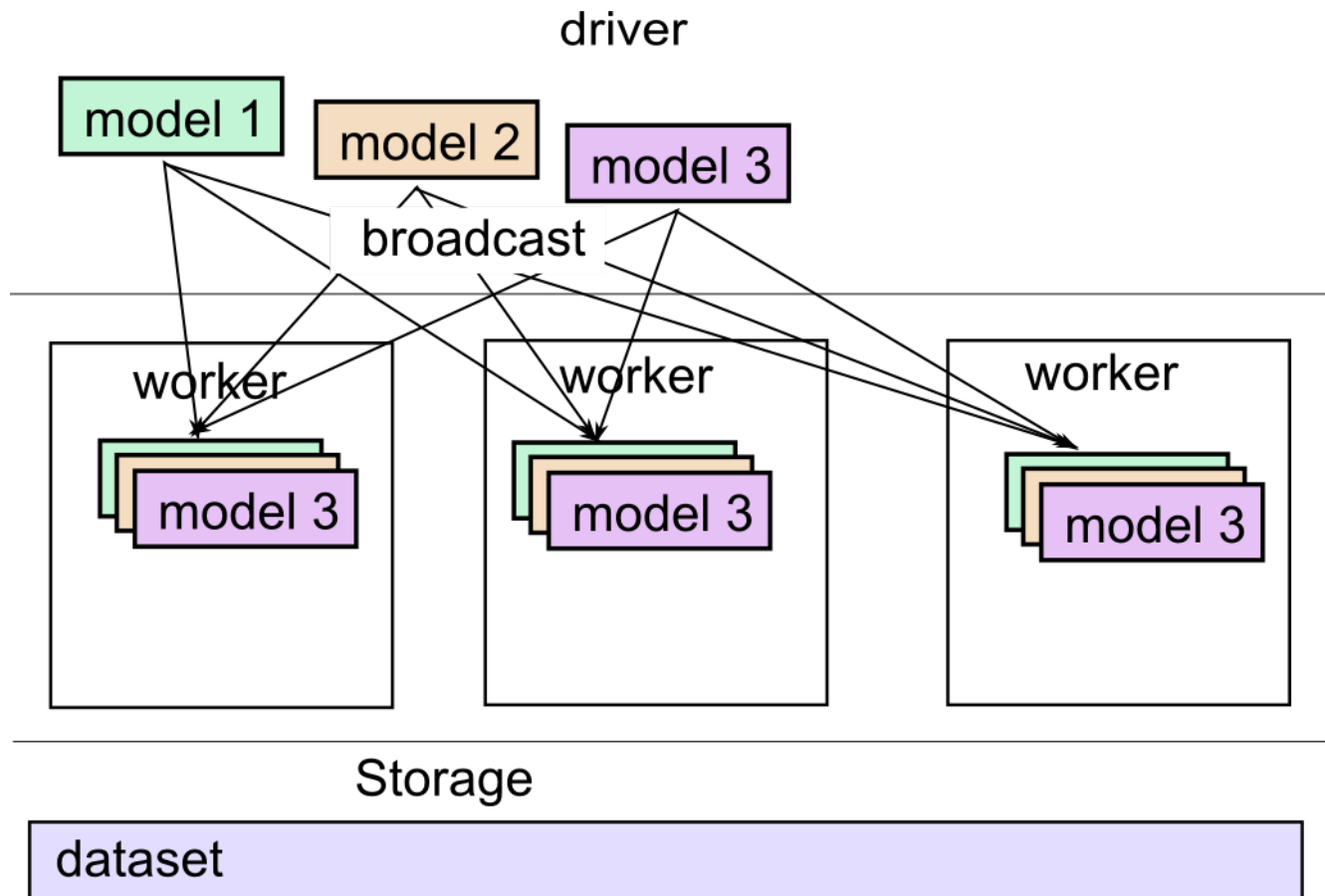
Storage



# Distributed prediction



# Distributed prediction



# Distributed prediction

