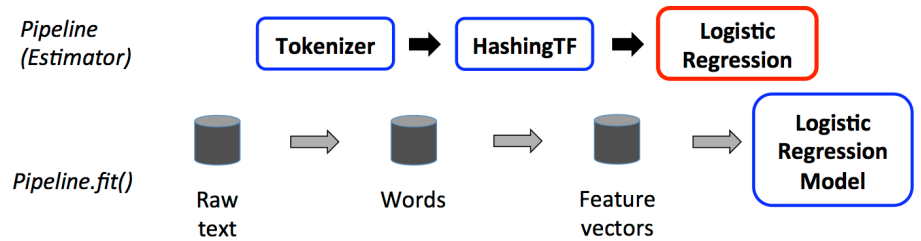
Model Pipelines in `MLlib`

Justin Post

`MLlib`

- Allows us to fit our ML models in Spark!
- Setting up response and predictors:
  - Create a `label` column which represents the response
  - Create a `features` column with all of the predictors in it!
- Many functions with a `.transform()` method
- Models and CV function have a `.fit()` method (once fitted a `.transform()` method too!)



CV in `MLlib`

- Similar to `GridSearchCV()` in `sklearn`, `CrossValidator()` allows us to do CV easily!
  - `ParamGridBuilder()` allows us to easily create a grid of tuning parameters
  - Set up `CrossValidator()` object and then use the `.fit()` method!

CV in `MLlib`

- Similar to `GridSearchCV()` in `sklearn`, `CrossValidator()` allows us to do CV easily!
  - `ParamGridBuilder()` allows us to easily create a grid of tuning parameters
  - Set up `CrossValidator()` object and then use the `.fit()` method!

```
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.evaluation import RegressionEvaluator
```

CV in `MLlib`

- Similar to `GridSearchCV()` in `sklearn`, `CrossValidator()` allows us to do CV easily!
  - `ParamGridBuilder()` allows us to easily create a grid of tuning parameters
  - Set up `CrossValidator()` object and then use the `.fit()` method!

```
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.evaluation import RegressionEvaluator

lr = LinearRegression()
paramGrid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0, 0.5]) \
    .addGrid(lr.elasticNetParam, [0, 0.2]) \
    .build()
```

CV in `MLlib`

- Similar to `GridSearchCV()` in `sklearn`, `CrossValidator()` allows us to do CV easily!
  - `ParamGridBuilder()` allows us to easily create a grid of tuning parameters
  - Set up `CrossValidator()` object and then use the `.fit()` method!

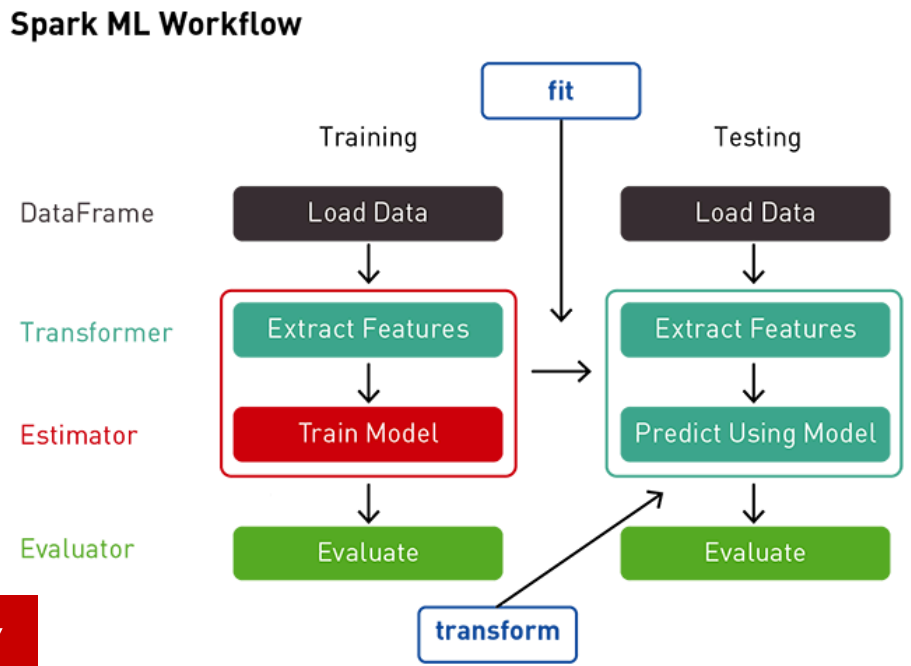
```
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.evaluation import RegressionEvaluator

lr = LinearRegression()
paramGrid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0, 0.5]) \
    .addGrid(lr.elasticNetParam, [0, 0.2]) \
    .build()
crossval = CrossValidator(estimator = lr,
                          estimatorParamMaps = paramGrid,
                          evaluator = RegressionEvaluator(metricName='rmse'),
                          numFolds=5) #now use .fit() method on crossval!
```

Pipelines

- Often have many data transformation steps
- Followed by fitting the model
- Then want to use the model to predict!

Pipelines

- Often have many data transformation steps
- Followed by fitting the model
- Then want to use the model to predict!



Pipelines

- Often have many data transformation steps
- Followed by fitting the model
- Then want to use the model to predict!
- `Pipeline()` allows us to wrap all this into an easy call

```
from pyspark.ml import Pipeline
pipeline = Pipeline(stages = [sqlTrans, assembler, lr])
crossval = CrossValidator(estimator = pipeline,
                          estimatorParamMaps = paramGrid,
                          evaluator = RegressionEvaluator(),
                          numFolds=5)
cvModel = crossval.fit(train)
cvModel.transform(test) #for predictions
```

To `pyspark`

Let's do quick example of CV and using Pipelines in `MLlib`!

Recap

- CV easy to do, similar to `sklearn`
- Pipelines make the process simpler
    - Easy to predict
    - Easy to switch model type