In [1]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-27-ML-regression")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']
```

In [3]:
```
df = spark.read.load(SPARK_BOOK_DATA_PATH + "/data/regression")


# COMMAND ----------

from pyspark.ml.regression import LinearRegression
lr = LinearRegression().setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)

lrModel = lr.fit(df)


# COMMAND ----------
print (lr.explainParams())
summary = lrModel.summary
summary.residuals.show()
summary_str = f"""
totalIterations: {summary.totalIterations}
objectiveHistory: {summary.objectiveHistory}
rootMeanSquaredError: {summary.rootMeanSquaredError}
r2: {summary.r2}
"""
print(summary_str)
```

```
aggregationDepth: suggested depth for treeAggregate (>= 2). (default: 2)
elasticNetParam: the ElasticNet mixing parameter, in range [0, 1]. For alpha = 0, the penalty is an
L2 penalty. For alpha = 1, it is an L1 penalty. (default: 0.0, current: 0.8)
epsilon: The shape parameter to control the amount of robustness. Must be > 1.0. Only valid when lo
ss is huber (default: 1.35)
featuresCol: features column name. (default: features)
fitIntercept: whether to fit an intercept term. (default: True)
labelCol: label column name. (default: label)
loss: The loss function to be optimized. Supported options: squaredError, huber. (default: squaredE
rror)
maxIter: max number of iterations (>= 0). (default: 100, current: 10)
predictionCol: prediction column name. (default: prediction)
regParam: regularization parameter (>= 0). (default: 0.0, current: 0.3)
solver: The solver algorithm for optimization. Supported options: auto, normal, l-bfgs. (default: a
uto)
standardization: whether to standardize the training features before fitting the model. (default: T
rue)
tol: the convergence tolerance for iterative algorithms (>= 0). (default: 1e-06)
weightCol: weight column name. If this is not set or empty, we treat all instance weights as 1.0.
(undefined)
```

```
+--------------------+
|           residuals|
+--------------------+
| 0.12805046585610125|
|-0.14468269261571942|
|-0.41903832622420634|
|-0.41903832622420634|
|  0.8547088792080308|
+--------------------+


totalIterations: 6
objectiveHistory: [0.5000000000000001, 0.43152958103627864, 0.313233593388102, 0.31225692666554095,
0.30915060819830315, 0.30915058933480255]
rootMeanSquaredError: 0.47308424392175996
r2: 0.7202391226912209
```

In [4]:
```
# COMMAND ----------

from pyspark.ml.regression import GeneralizedLinearRegression
glr = GeneralizedLinearRegression()\
  .setFamily("gaussian")\
  .setLink("identity")\
  .setMaxIter(10)\
  .setRegParam(0.3)\
  .setLinkPredictionCol("linkOut")

glrModel = glr.fit(df)
```

In [7]:
```
# COMMAND ----------
print (glr.explainParams())
summary = glrModel.summary
```

```
family: The name of family which is a description of the error distribution to be used in the mode
l. Supported options: gaussian (default), binomial, poisson, gamma and tweedie. (default: gaussian,
current: gaussian)
featuresCol: features column name. (default: features)
fitIntercept: whether to fit an intercept term. (default: True)
labelCol: label column name. (default: label)
link: The name of link function which provides the relationship between the linear predictor and th
e mean of the distribution function. Supported options: identity, log, inverse, logit, probit, clog
log and sqrt. (current: identity)
linkPower: The index in the power link function. Only applicable to the Tweedie family. (undefined)
linkPredictionCol: link prediction (linear predictor) column name (current: linkOut)
maxIter: max number of iterations (>= 0). (default: 25, current: 10)
offsetCol: The offset column name. If this is not set or empty, we treat all instance offsets as 0.
0 (undefined)
predictionCol: prediction column name. (default: prediction)
regParam: regularization parameter (>= 0). (default: 0.0, current: 0.3)
solver: The solver algorithm for optimization. Supported options: irls. (default: irls)
tol: the convergence tolerance for iterative algorithms (>= 0). (default: 1e-06)
variancePower: The power in the variance function of the Tweedie distribution which characterizes t
he relationship between the variance and mean of the distribution. Only applicable for the Tweedie
family. Supported values: 0 and [1, Inf). (default: 0.0)
weightCol: weight column name. If this is not set or empty, we treat all instance weights as 1.0.
(undefined)
```

In [9]:
```
summary.numIterations
```

Out[9]: 1

In [10]:
```
summary.residuals
```

Out[10]:
```
<bound method GeneralizedLinearRegressionSummary.residuals of Coefficients:
    Feature Estimate Std Error T Value P Value
(Intercept)   0.0867    1.2210  0.0710  0.9549
 features_0   0.3661    0.7686  0.4764  0.7170
 features_1   0.0466    0.1380  0.3377  0.7927
 features_2   0.1831    0.3843  0.4764  0.7170

(Dispersion parameter for gaussian family taken to be 0.8466)
    Null deviance: 4.0000 on 1 degrees of freedom
Residual deviance: 0.8466 on 1 degrees of freedom
AIC: 15.3094>
```

In [11]:
```
# COMMAND ----------

from pyspark.ml.regression import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
print (dtr.explainParams())
dtrModel = dtr.fit(df)
```

```
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes. I
f true, the algorithm will cache node IDs for each instance. Caching can speed up training of deepe
r trees. Users can set how often should the cache be checkpointed or disable it by setting checkpoi
ntInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that t
he cache will get checkpointed every 10 iterations. Note: this setting will be ignored if the check
point directory is not set in the SparkContext. (default: 10)
featuresCol: features column name. (default: features)
impurity: Criterion used for information gain calculation (case-insensitive). Supported options: va
riance (default: variance)
labelCol: label column name. (default: label)
maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of cat
egories for any categorical feature. (default: 32)
maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 intern
al node + 2 leaf nodes. (default: 5)
maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node w
ill be split per iteration, and its aggregates may exceed this size. (default: 256)
minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
minInstancesPerNode: Minimum number of instances each child must have after split. If a split cause
s the left or right child to have fewer than minInstancesPerNode, the split will be discarded as in
valid. Should be >= 1. (default: 1)
predictionCol: prediction column name. (default: prediction)
seed: random seed. (default: -1407754390808368278)
varianceCol: column name for the biased sample variance of prediction. (undefined)
```

In [12]:
```
# COMMAND ----------

from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.regression import GBTRegressor
rf =  RandomForestRegressor()
print (rf.explainParams())
rfModel = rf.fit(df)
```

```
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes. I
f true, the algorithm will cache node IDs for each instance. Caching can speed up training of deepe
r trees. Users can set how often should the cache be checkpointed or disable it by setting checkpoi
ntInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that t
he cache will get checkpointed every 10 iterations. Note: this setting will be ignored if the check
point directory is not set in the SparkContext. (default: 10)
featureSubsetStrategy: The number of features to consider for splits at each tree node. Supported o
ptions: 'auto' (choose automatically for task: If numTrees == 1, set to 'all'. If numTrees > 1 (for
est), set to 'sqrt' for classification and to 'onethird' for regression), 'all' (use all features),
'onethird' (use 1/3 of the features), 'sqrt' (use sqrt(number of features)), 'log2' (use log2(numbe
r of features)), 'n' (when n is in the range (0, 1.0], use n * number of features. When n is in the
range (1, number of features), use n features). default = 'auto' (default: auto)
featuresCol: features column name. (default: features)
impurity: Criterion used for information gain calculation (case-insensitive). Supported options: va
riance (default: variance)
labelCol: label column name. (default: label)
maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of cat
egories for any categorical feature. (default: 32)
maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 intern
al node + 2 leaf nodes. (default: 5)
maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node w
ill be split per iteration, and its aggregates may exceed this size. (default: 256)
minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
minInstancesPerNode: Minimum number of instances each child must have after split. If a split cause
s the left or right child to have fewer than minInstancesPerNode, the split will be discarded as in
valid. Should be >= 1. (default: 1)
numTrees: Number of trees to train (>= 1). (default: 20)
predictionCol: prediction column name. (default: prediction)
seed: random seed. (default: 2502083311556356884)
subsamplingRate: Fraction of the training data used for learning each decision tree, in range (0,
1]. (default: 1.0)
```

In [13]:
```
gbt = GBTRegressor()
print (gbt.explainParams())
gbtModel = gbt.fit(df)
```

```
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes. I
f true, the algorithm will cache node IDs for each instance. Caching can speed up training of deepe
r trees. Users can set how often should the cache be checkpointed or disable it by setting checkpoi
ntInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that t
he cache will get checkpointed every 10 iterations. Note: this setting will be ignored if the check
point directory is not set in the SparkContext. (default: 10)
featureSubsetStrategy: The number of features to consider for splits at each tree node. Supported o
ptions: 'auto' (choose automatically for task: If numTrees == 1, set to 'all'. If numTrees > 1 (for
est), set to 'sqrt' for classification and to 'onethird' for regression), 'all' (use all features),
'onethird' (use 1/3 of the features), 'sqrt' (use sqrt(number of features)), 'log2' (use log2(numbe
r of features)), 'n' (when n is in the range (0, 1.0], use n * number of features. When n is in the
range (1, number of features), use n features). default = 'auto' (default: all)
featuresCol: features column name. (default: features)
impurity: Criterion used for information gain calculation (case-insensitive). Supported options: va
riance (default: variance)
labelCol: label column name. (default: label)
lossType: Loss function which GBT tries to minimize (case-insensitive). Supported options: squared,
absolute (default: squared)
maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of cat
egories for any categorical feature. (default: 32)
maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 intern
al node + 2 leaf nodes. (default: 5)
maxIter: max number of iterations (>= 0). (default: 20)
maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node w
ill be split per iteration, and its aggregates may exceed this size. (default: 256)
minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
minInstancesPerNode: Minimum number of instances each child must have after split. If a split cause
s the left or right child to have fewer than minInstancesPerNode, the split will be discarded as in
valid. Should be >= 1. (default: 1)
predictionCol: prediction column name. (default: prediction)
seed: random seed. (default: -6682481135904123338)
stepSize: Step size (a.k.a. learning rate) in interval (0, 1] for shrinking the contribution of eac
h estimator. (default: 0.1)
subsamplingRate: Fraction of the training data used for learning each decision tree, in range (0,
1]. (default: 1.0)
```

In [15]:
```
# COMMAND ----------

from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.regression import GeneralizedLinearRegression
from pyspark.ml import Pipeline
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
glr = GeneralizedLinearRegression().setFamily("gaussian").setLink("identity")

pipeline = Pipeline().setStages([glr])

params = ParamGridBuilder()\
    .addGrid(glr.regParam, [0, 0.5, 1])\
    .build()

evaluator = RegressionEvaluator()\
  .setMetricName("rmse")\
  .setPredictionCol("prediction")\
  .setLabelCol("label")

cv = CrossValidator()\
  .setEstimator(pipeline)\
  .setEvaluator(evaluator)\
  .setEstimatorParamMaps(params)\
  .setNumFolds(2) # should always be 3 or more but this dataset is small

model = cv.fit(df)
```

In [16]:
```
# COMMAND ----------

from pyspark.mllib.evaluation import RegressionMetrics
out = model.transform(df)\
  .select("prediction", "label").rdd.map(lambda x: (float(x[0]), float(x[1])))
metrics = RegressionMetrics(out)
print ("MSE: " + str(metrics.meanSquaredError))
print ("RMSE: " + str(metrics.rootMeanSquaredError))
print ("R-squared: " + str(metrics.r2))
print ("MAE: " + str(metrics.meanAbsoluteError))
print ("Explained variance: " + str(metrics.explainedVariance))


# COMMAND ----------
```

```
MSE: 0.15705521472392636
RMSE: 0.39630192369445594
R-squared: 0.803680981595092
MAE: 0.31411042944785267
Explained variance: 0.6429447852760728
```

In [ ]:
```

```