Creación Objeto Spark Session.

FINISHED

```
import org.apache.spark.sql.SparkSession

val spark = SparkSession.builder.appName("Ames").appName("appSample")
                                                .getOrCreate()
```

```
import org.apache.spark.sql.SparkSession
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@682f6b64
```

Took 0 sec. Last updated by anonymous at September 06 2019, 12:53:42 PM. (outdated)

Carga de Data Sets

FINISHED

```
val train = spark.read.format("csv")
                      .option("header", "true")
                      .option("mode", "FAILFAST")
                      .option("inferSchema", "true")
                      .load("s3://aws-logs-079715545706-eu-west-3/X_train_embedd.csv")

val test = spark.read.format("csv")
                      .option("header", "true")
                      .option("mode", "FAILFAST")
                      .option("inferSchema", "true")
                      .load("s3://aws-logs-079715545706-eu-west-3/X_test_embedd.csv")
```

```
train: org.apache.spark.sql.DataFrame = [SalePrice: double, MSSubClass_70: int ... 58 more fields]
test: org.apache.spark.sql.DataFrame = [MSSubClass_70: int, MSZoning_RM: int ... 57 more fields]
```

Took 12 sec. Last updated by anonymous at September 06 2019, 11:12:12 AM. (outdated)

Librerías necesarias

FINISHED

```
import org.apache.spark.ml.feature.RFormula

import org.apache.spark.ml.tuning.{ParamGridBuilder, CrossValidator}
import org.apache.spark.ml.evaluation.RegressionEvaluator

import org.apache.spark.ml.regression.{LinearRegression, IsotonicRegression, RandomForestRegressor, GBTRegressor}

val cv = 10
```

Creación de modelo

FINISHED

```
val model = new RFormula().setFormula("SalePrice ~ .").fit(train).transform(train)
```

Ridge Regression

FINISHED

```
val lr_ridge = new LinearRegression().setFeaturesCol("features").setLabelCol("label")
```

FINISHED

```
val lr_ridge_param = new ParamGridBuilder().addGrid(lr_ridge.elasticNetParam, Array(0.0))
                                           .addGrid(lr_ridge.regParam, (0.0001 to 1 by 0.001).toArray)
                                           .build()

val lr_ridge_score = new RegressionEvaluator().setMetricName("rmse").setLabelCol("label").setPredictionCol("prediction")

val lr_ridge_model = new CrossValidator().setEstimator(lr_ridge)
                                         .setEvaluator(lr_ridge_score)
                                         .setEstimatorParamMaps(lr_ridge_param)
                                         .setNumFolds(cv)
```

FINISHED

```
val lr_ridge_modelfit = lr_ridge_model.fit(model)
```

FINISHED

```
lr_ridge_modelfit.bestModel.extractParamMap()
```

amesSpark

```
res1: org.apache.spark.ml.param.ParamMap =
{
        linReg_774107292ad3-aggregationDepth: 2,
        linReg_774107292ad3-elasticNetParam: 0.0,
        linReg_774107292ad3-epsilon: 1.35,
        linReg_774107292ad3-featuresCol: features,
        linReg_774107292ad3-fitIntercept: true,
        linReg_774107292ad3-labelCol: label,
        linReg_774107292ad3-loss: squaredError,
        linReg_774107292ad3-maxIter: 100,
        linReg_774107292ad3-predictionCol: prediction,
        linReg_774107292ad3-regParam: 1.0E-4,
        linReg_774107292ad3-solver: auto,
        linReg_774107292ad3-standardization: true,
        linReg_774107292ad3-tol: 1.0E-6
}
```

Took 1 sec. Last updated by anonymous at September 06 2019, 9:43:14 AM.

```
lr_ridge_modelfit.avgMetrics
```

FINISHED

```
res2: Array[Double] = Array(0.11233397434522183, 0.11314348305502175, 0.11351871698365419, 0.11377343045579122, 0.11402854635479359, 0.11430873714698958,
0.11461692735967993, 0.11495061526048014, 0.11530620200945016, 0.11568022732228261, 0.11606968994206596, 0.11647208133662607, 0.11688533339207792, 0.117307
74742987009, 0.11773792674036235, 0.11817471889459424, 0.11861716848656645, 0.11906447914005756, 0.11951598319816073, 0.11997111759125459, 0.12042940460539
04, 0.1208904365148093, 0.121353863258025, 0.12181938251418231, 0.1222867316774341, 0.1227556813375485, 0.12322602996072986, 0.12369759953073106, 0.1241702
3196159419, 0.12464378613335733, 0.12511813543241737, 0.12559316570291085, 0.1260687735336096, 0.12654486482048388, 0.12702135355590524, 0.127498160805379
1, 0.1279752138397142, 0...
```

Took 1 sec. Last updated by anonymous at September 06 2019, 9:43:15 AM.

Lasso Regression

FINISHED

```
val lr_lasso = new LinearRegression().setFeaturesCol("features").setLabelCol("label")
```

FINISHED

```
val lr_lasso_param = new ParamGridBuilder().addGrid(lr_lasso.elasticNetParam, Array(1.0))
                                           .addGrid(lr_lasso.regParam, (0.0001 to 1 by 0.001).toArray)
                                           .build()

val lr_lasso_score = new RegressionEvaluator().setMetricName("rmse").setLabelCol("label").setPredictionCol("prediction")

val lr_lasso_model = new CrossValidator().setEstimator(lr_lasso)
                                         .setEvaluator(lr_lasso_score)
                                         .setEstimatorParamMaps(lr_lasso_param)
                                         .setNumFolds(cv)
```

FINISHED

```
val lr_lasso_modelfit = lr_lasso_model.fit(model)
```

```
lr_lasso_modelfit: org.apache.spark.ml.tuning.CrossValidatorModel = cv_fb0931c687a2
```

Took 1 min 18 sec. Last updated by anonymous at September 06 2019, 9:44:34 AM.

FINISHED

```
lr_lasso_modelfit.bestModel.extractParamMap()
```

```
res3: org.apache.spark.ml.param.ParamMap =
{
        linReg_c3e6d2e7a5ce-aggregationDepth: 2,
        linReg_c3e6d2e7a5ce-elasticNetParam: 1.0,
        linReg_c3e6d2e7a5ce-epsilon: 1.35,
        linReg_c3e6d2e7a5ce-featuresCol: features,
        linReg_c3e6d2e7a5ce-fitIntercept: true,
        linReg_c3e6d2e7a5ce-labelCol: label,
        linReg_c3e6d2e7a5ce-loss: squaredError,
        linReg_c3e6d2e7a5ce-maxIter: 100,
        linReg_c3e6d2e7a5ce-predictionCol: prediction,
        linReg_c3e6d2e7a5ce-regParam: 1.0E-4,
        linReg_c3e6d2e7a5ce-solver: auto,
        linReg_c3e6d2e7a5ce-standardization: true,
        linReg_c3e6d2e7a5ce-tol: 1.0E-6
}
```

Took 0 sec. Last updated by anonymous at September 06 2019, 9:44:34 AM.

```
lr_lasso_modelfit.avgMetrics
```

FINISHED

```
res4: Array[Double] = Array(0.11225569931971979, 0.12296985871329909, 0.1330693286459863, 0.14183793632355313, 0.14948598596430368, 0.1572819740619554, 0.1
6545350678520465, 0.17353063541112773, 0.18128440373014437, 0.1892386959608343, 0.1975214126581231, 0.20607049637940603, 0.2150081335696463, 0.224117401491
43908, 0.23296486010749284, 0.24196502140563309, 0.25123025515060277, 0.2607323408367709, 0.270316498060075, 0.27998509263246074, 0.2898268853214553, 0.299
69623529868605, 0.3094030509726674, 0.3188336612375139, 0.32838028894026083, 0.3380454778507471, 0.3472378443138924, 0.35573647861320135, 0.363483417446434
2, 0.37123955889016413, 0.3791013487491578, 0.3870061762908954, 0.39077260884008236, 0.39458935068353185, 0.39845368847487694, 0.3997825083284514, 0.399782
5083284514, 0.39978250...
```

Took 0 sec. Last updated by anonymous at September 06 2019, 9:44:34 AM.

amesSpark

ElasticNet Regression

FINISHED

```
val lr_elasNet = new LinearRegression().setFeaturesCol("features").setLabelCol("label")
```

```
lr_elasNet: org.apache.spark.ml.regression.LinearRegression = linReg_0e89c6fe4350
```

Took 0 sec. Last updated by anonymous at September 06 2019, 11:13:36 AM.

FINISHED

```
val lr_elasNet_param = new ParamGridBuilder().addGrid(lr_elasNet.elasticNetParam, (0.1 to 1 by 0.1).toArray)
                                         .addGrid(lr_elasNet.regParam,  (0.0001 to 1 by 0.01).toArray)
                                         .build()

val lr_elasNet_score = new RegressionEvaluator().setMetricName("rmse").setLabelCol("label").setPredictionCol("prediction")

val lr_elasNet_model = new CrossValidator().setEstimator(lr_elasNet)
                                       .setEvaluator(lr_elasNet_score)
                                       .setEstimatorParamMaps(lr_elasNet_param)
                                       .setNumFolds(cv)
```

FINISHED

```
val lr_elasNet_modelfit = lr_elasNet_model.fit(model)
```

```
lr_elasNet_modelfit: org.apache.spark.ml.tuning.CrossValidatorModel = cv_dce977e0b63d
```

Took 11 min 37 sec. Last updated by anonymous at September 06 2019, 11:25:25 AM.

FINISHED

```
lr_elasNet_modelfit.bestModel.extractParamMap()
```

```
res3: org.apache.spark.ml.param.ParamMap =
{
        linReg_0e89c6fe4350-aggregationDepth: 2,
        linReg_0e89c6fe4350-elasticNetParam: 1.0,
        linReg_0e89c6fe4350-epsilon: 1.35,
        linReg_0e89c6fe4350-featuresCol: features,
        linReg_0e89c6fe4350-fitIntercept: true,
        linReg_0e89c6fe4350-labelCol: label,
        linReg_0e89c6fe4350-loss: squaredError,
        linReg_0e89c6fe4350-maxIter: 100,
        linReg_0e89c6fe4350-predictionCol: prediction,
        linReg_0e89c6fe4350-regParam: 1.0E-4,
        linReg_0e89c6fe4350-solver: auto,
        linReg_0e89c6fe4350-standardization: true,
        linReg_0e89c6fe4350-tol: 1.0E-6
}
```

Took 0 sec. Last updated by anonymous at September 06 2019, 11:29:54 AM.

FINISHED

```
lr_elasNet_modelfit.avgMetrics
```

```
res4: Array[Double] = Array(0.11239225468324407, 0.11396039474960551, 0.11469327316859887, 0.11551916506172974, 0.1165505069306456, 0.11775669949932221, 0.
11909274816817776, 0.12050136153444911, 0.12204103377309566, 0.12358981517992854, 0.12516238345813635, 0.12677636135973727, 0.12828546530234464, 0.12973067
10721668, 0.13119830592852777, 0.1326934072690988, 0.1341431469164818, 0.13554341828959185, 0.1369255263543971, 0.13833460959830998, 0.13974425027793103,
0.1411270700438802, 0.14253252990441406, 0.14395903689787914, 0.14540493999666382, 0.14686768694454289, 0.14834503972807805, 0.14982663564697413, 0.1512830
5242722997, 0.15264898973017885, 0.15401549658931207, 0.15539578880929011, 0.15678895375576668, 0.15817845208707415, 0.15949954683381878, 0.160813219231845
94, 0.16211434859940224...
```

Took 0 sec. Last updated by anonymous at September 06 2019, 11:29:59 AM.

Isotonic Regression

FINISHED

```
import org.apache.spark.ml.regression.IsotonicRegression
```

```
import org.apache.spark.ml.regression.IsotonicRegression
```

Took 0 sec. Last updated by anonymous at September 06 2019, 9:55:44 AM.

FINISHED

```
val lr_iso = new IsotonicRegression().setFeaturesCol("features").setLabelCol("label")
```

```
lr_iso: org.apache.spark.ml.regression.IsotonicRegression = isoReg_11fc69ef02cf
```

Took 1 sec. Last updated by anonymous at September 06 2019, 9:55:45 AM.

FINISHED

```
val lr_iso_param = new ParamGridBuilder().addGrid(lr_iso.isotonic, Array(true, false))
                                         .build()

val lr_iso_score = new RegressionEvaluator().setMetricName("rmse").setLabelCol("label").setPredictionCol("prediction")

val lr_iso_model = new CrossValidator().setEstimator(lr_iso)
                                       .setEvaluator(lr_iso_score)
                                       .setEstimatorParamMaps(lr_iso_param)
                                       .setNumFolds(cv)
```

amesSpark

```
val lr_iso_modelfit = lr_iso_model.fit(model)
```

FINISHED

```
lr_iso_modelfit: org.apache.spark.ml.tuning.CrossValidatorModel = cv_92d7adcbc79b
```

Took 5 sec. Last updated by anonymous at September 06 2019, 9:55:50 AM.

```
lr_iso_modelfit.bestModel.extractParamMap()
```

FINISHED

```
lr_iso_modelfit.avgMetrics
```

FINISHED

```
res8: Array[Double] = Array(0.4177412416241565, 0.41973045976494294)
```

Took 1 sec. Last updated by anonymous at September 06 2019, 9:55:51 AM.

Random Forest Regressor

FINISHED

```
val rf = new RandomForestRegressor().setFeaturesCol("features").setLabelCol("label").setSeed(12345)
```

```
rf: org.apache.spark.ml.regression.RandomForestRegressor = rfr_f832856f7c1b
```

Took 0 sec. Last updated by anonymous at September 06 2019, 11:32:01 AM.

FINISHED

```
val rf_param = new ParamGridBuilder().addGrid(rf.maxDepth, Array(10, 20, 30, 40, 50))
                                     .addGrid(rf.numTrees, (100 to 1000 by 100).toArray)
                                     .addGrid(rf.numTrees, (100 to 1000 by 100).toArray)
                                     .addGrid(rf.featureSubsetStrategy, Array("all", "sqrt", "log2"))
                                     .build()

val rf_score = new RegressionEvaluator().setMetricName("rmse").setLabelCol("label").setPredictionCol("prediction")

val rf_model = new CrossValidator().setEstimator(rf)
                                   .setEvaluator(rf_score)
                                   .setEstimatorParamMaps(rf_param)
                                   .setNumFolds(cv)
```

FINISHED

```
val rf_modelfit = rf_model.fit(model)
```

```
rf_modelfit: org.apache.spark.ml.tuning.CrossValidatorModel = cv_d31eefa71391
```

Took 45 sec. Last updated by anonymous at September 06 2019, 11:33:02 AM.

FINISHED

```
rf_modelfit.bestModel.extractParamMap()
```

```
res5: org.apache.spark.ml.param.ParamMap =
{
        rfr_f832856f7c1b-cacheNodeIds: false,
        rfr_f832856f7c1b-checkpointInterval: 10,
        rfr_f832856f7c1b-featureSubsetStrategy: all,
        rfr_f832856f7c1b-featuresCol: features,
        rfr_f832856f7c1b-impurity: variance,
        rfr_f832856f7c1b-labelCol: label,
        rfr_f832856f7c1b-maxBins: 32,
        rfr_f832856f7c1b-maxDepth: 5,
        rfr_f832856f7c1b-maxMemoryInMB: 256,
        rfr_f832856f7c1b-minInfoGain: 0.0,
        rfr_f832856f7c1b-minInstancesPerNode: 1,
        rfr_f832856f7c1b-numTrees: 400,
        rfr_f832856f7c1b-predictionCol: prediction,
        rfr_f832856f7c1b-seed: 12345,
        rfr_f832856f7c1b-subsamplingRate: 1.0
}
```

Took 1 sec. Last updated by anonymous at September 06 2019, 11:33:18 AM.

Gradient Boosting Regressor

FINISHED

```
val gbm = new GBTRegressor().setFeaturesCol("features").setLabelCol("label").setSeed(12345)
```

```
gbm: org.apache.spark.ml.regression.GBTRegressor = gbtr_e14f91652137
```

Took 0 sec. Last updated by anonymous at September 06 2019, 11:35:01 AM.

FINISHED

amesSpark

```
val gbm_param = new ParamGridBuilder().addGrid(gbm.maxDepth, Array(5, 10, 15))
                                      .addGrid(gbm.featureSubsetStrategy, Array("all", "sqrt", "log2"))
                                      .addGrid(gbm.stepSize, (0.001 to 1 by 0.1).toArray)
                                      .addGrid(gbm.minInstancesPerNode, (10 to 100 by 10).toArray)
                                      .addGrid(gbm.subsamplingRate, Array(0.7, 0.8, 0.9, 1))
                                      .addGrid(gbm.lossType, Array("squared", "absolute"))
                                      .build()
```

```
val gbm_score = new RegressionEvaluator().setMetricName("rmse").setLabelCol("label").setPredictionCol("prediction")

val gbm_model = new CrossValidator().setEstimator(gbm)
                                    .setEvaluator(gbm_score)
                                    .setEstimatorParamMaps(gbm_param)
                                    .setNumFolds(cv)
```

```
val gbm_modelfit = gbm_model.fit(model)
```

FINISHED

```
gbm_modelfit: org.apache.spark.ml.tuning.CrossValidatorModel = cv_8fc2b6eb287e
```

Took 40 min 45 sec. Last updated by anonymous at September 06 2019, 12:36:53 PM.

```
gbm_modelfit.bestModel.extractParamMap()
```

FINISHED

```
gbm_modelfit.avgMetrics
```

FINISHED

```
res16: Array[Double] = Array(0.19601147816171358, 0.18146048547920757, 0.18758598059882164, 0.1853358870394609, 0.1938639202602427, 0.20170636000502462, 0.
21953708435224786, 0.24192836143818086, 0.27707684230124174, 0.3106681614575216, 0.20656260763325357, 0.17459721059313194, 0.17502435052770432, 0.180623794
15771437, 0.19444625058789677, 0.2082721632856439, 0.24056401225942625, 0.2614909221172763, 0.3144032301749212, 0.3586198159939865, 0.23004802601445415, 0.
17540383483544658, 0.17387610684422805, 0.1852788769314995, 0.19609952886220894, 0.21379733082451202, 0.23509984142147566, 0.26530349563789213, 0.320864770
40457007, 0.40488915077790366, 0.1989883484507789, 0.1970578474593382, 0.1978833376382873, 0.19863578652031666, 0.20001051589182112, 0.20053426596732968,
0.20384272110585894, 0....
```

Took 0 sec. Last updated by anonymous at September 06 2019, 12:41:51 PM.

READY