Chapter 8: Tree Models

Ex3: KDD 10%

Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between 'bad' connections, called intrusions or attacks, and 'good' normal connections.

Read more: [https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data (https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data)](https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data)

Requirement: We will be using a KDD dataset to try to classify a connection as 'normal.' or others.

In [1]:
```
import findspark
findspark.init()
```

In [2]:
```
from pyspark.sql import SparkSession
```

In [3]:
```
spark = SparkSession.builder.appName('kdd').getOrCreate()
```

In [4]:
```
df = spark.read.csv('kdd_data/kddcup.data_10_percent.gz',
                    inferSchema=True, header=False)
```

In [5]:
```
df.count()
```

Out[5]:
```
494021
```

In [9]:
```
from pyspark.sql.functions import col
```

In [10]:
```
str(df.columns)
```

Out[10]:
```
"['_c0', '_c1', '_c2', '_c3', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9', '_c10',
'_c11', '_c12', '_c13', '_c14', '_c15', '_c16', '_c17', '_c18', '_c19', '_c20',
'_c21', '_c22', '_c23', '_c24', '_c25', '_c26', '_c27', '_c28', '_c29', '_c30',
'_c31', '_c32', '_c33', '_c34', '_c35', '_c36', '_c37', '_c38', '_c39', '_c40',
'_c41']"
```

In [11]:
```
cols = ['_c0', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9', '_c10',
        '_c11', '_c12', '_c13', '_c14', '_c15', '_c16', '_c17',
        '_c18', '_c19', '_c20', '_c21', '_c22', '_c23', '_c24',
        '_c25', '_c26', '_c27', '_c28', '_c29', '_c30', '_c31',
        '_c32', '_c33', '_c34', '_c35', '_c36', '_c37', '_c38',
        '_c39', '_c40']
for col_name in cols:
    df = df.withColumn(col_name, col(col_name).cast('float'))
```

In [12]:
```
df.show(3)
```

```
+---+---+----+---+-----+------+---+---+---+---+----+----+----+----+----+----+--
--+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-
---+----+----+----+----+----+----+----+----+-------+
|_c0|_c1| _c2|_c3|  _c4|   _c5|_c6|_c7|_c8|_c9|_c10|_c11|_c12|_c13|_c14|_c15|_c
16|_c17|_c18|_c19|_c20|_c21|_c22|_c23|_c24|_c25|_c26|_c27|_c28|_c29|_c30|_c31|_
c32|_c33|_c34|_c35|_c36|_c37|_c38|_c39|_c40|   _c41|
+---+---+----+---+-----+------+---+---+---+---+----+----+----+----+----+----+--
--+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-
---+----+----+----+----+----+----+----+----+-------+
|0.0|tcp|http| SF|181.0|5450.0|0.0|0.0|0.0|0.0| 0.0| 1.0| 0.0| 0.0| 0.0| 0.0|
0.0| 0.0| 0.0| 0.0| 0.0| 0.0| 8.0| 8.0| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0| 9.0|
9.0| 1.0| 0.0|0.11| 0.0| 0.0| 0.0| 0.0|normal.|
|0.0|tcp|http| SF|239.0| 486.0|0.0|0.0|0.0|0.0| 0.0| 1.0| 0.0| 0.0| 0.0| 0.0|
0.0| 0.0| 0.0| 0.0| 0.0| 0.0| 8.0| 8.0| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|19.0|
19.0| 1.0| 0.0|0.05| 0.0| 0.0| 0.0| 0.0|normal.|
|0.0|tcp|http| SF|235.0|1337.0|0.0|0.0|0.0|0.0| 0.0| 1.0| 0.0| 0.0| 0.0| 0.0|
0.0| 0.0| 0.0| 0.0| 0.0| 0.0| 8.0| 8.0| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|29.0|
29.0| 1.0| 0.0|0.03| 0.0| 0.0| 0.0| 0.0|normal.|
+---+---+----+---+-----+------+---+---+---+---+----+----+----+----+----+----+--
--+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-
---+----+----+----+----+----+----+----+----+-------+
only showing top 3 rows
```

In [13]:
```
df.groupBy('_c41').count().show(30)
```

```
+----------------+------+
|            _c41| count|
+----------------+------+
|    warezmaster.|    20|
|          smurf.|280790|
|            pod.|   264|
|           imap.|    12|
|           nmap.|   231|
|   guess_passwd.|    53|
|        ipsweep.|  1247|
|      portsweep.|  1040|
|          satan.|  1589|
|           land.|    21|
|     loadmodule.|     9|
|      ftp_write.|     8|
|buffer_overflow.|    30|
|        rootkit.|    10|
|    warezclient.|  1020|
|       teardrop.|   979|
|           perl.|     3|
|            phf.|     4|
|       multihop.|     7|
|        neptune.|107201|
|           back.|  2203|
|            spy.|     2|
|         normal.| 97278|
+----------------+------+
```

In [14]:
```
str(df.columns)
```

Out[14]:
```
"['_c0', '_c1', '_c2', '_c3', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9', '_c10',
'_c11', '_c12', '_c13', '_c14', '_c15', '_c16', '_c17', '_c18', '_c19', '_c20',
'_c21', '_c22', '_c23', '_c24', '_c25', '_c26', '_c27', '_c28', '_c29', '_c30',
'_c31', '_c32', '_c33', '_c34', '_c35', '_c36', '_c37', '_c38', '_c39', '_c40',
'_c41']"
```

In [15]:
```
train_data,test_data = df.randomSplit([0.8,0.2])
```

In [16]:
```
test_data .groupBy('_c41').count().show(30)
```

```
+----------------+-----+
|            _c41|count|
+----------------+-----+
|    warezmaster.|    1|
|          smurf.|56131|
|            pod.|   68|
|           nmap.|   37|
|           imap.|    2|
|   guess_passwd.|    9|
|        ipsweep.|  251|
|      portsweep.|  208|
|          satan.|  337|
|           land.|    1|
|     loadmodule.|    3|
|buffer_overflow.|    9|
|    warezclient.|  196|
|       teardrop.|  162|
|           perl.|    1|
|       multihop.|    1|
|        neptune.|21600|
|           back.|  455|
|         normal.|19358|
+----------------+-----+
```

In [17]:
```
from pyspark.ml.feature import StringIndexer, OneHotEncoderEstimator
```

In [18]:
```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

In [19]:
```
# Import class for creating a pipeline
from pyspark.ml import Pipeline
```

In [20]:
```
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.classification import GBTClassifier, RandomForestClassifier
```

In [21]:
```
# Convert categorical strings to index values
indexer1 = StringIndexer(inputCol='_c1', outputCol='c1_idx')
indexer2 = StringIndexer(inputCol='_c2', outputCol='c2_idx')
indexer3 = StringIndexer(inputCol='_c3', outputCol='c3_idx')
indexer41 = StringIndexer(inputCol='_c41', outputCol='c41_idx')

# One-hot encode index values
onehot = OneHotEncoderEstimator(
    inputCols=['c1_idx', 'c2_idx', 'c3_idx'],
    outputCols=['c1_dummy', 'c2_dummy', 'c3_dummy']
)

# Assemble predictors into a single column
assembler = VectorAssembler(inputCols=['_c0', 'c1_dummy',
                                       'c2_dummy', 'c3_dummy',
                                       '_c4', '_c5', '_c6',
                                       '_c7', '_c8', '_c9',
                                       '_c10', '_c11', '_c12',
                                       '_c13', '_c14',
                                       '_c15', '_c16', '_c17',
                                       '_c18', '_c19',
                                       '_c20', '_c21', '_c22',
                                       '_c23', '_c24',
                                       '_c25', '_c26', '_c27',
                                       '_c28', '_c29',
                                       '_c30', '_c31', '_c32',
                                       '_c33', '_c34',
                                       '_c35', '_c36', '_c37',
                                       '_c38', '_c39', '_c40'],
                           outputCol='features')

# A linear regression object
dtc = DecisionTreeClassifier(featuresCol='features',
                      labelCol='c41_idx',
                      predictionCol='prediction')
```

In [22]:
```
# Construct a pipeline
pipeline = Pipeline(stages=[indexer1, indexer2, indexer3,
                            indexer41, onehot, assembler, dtc])
```

In [23]:
```
# Train the pipeline on the training data
pipeline = pipeline.fit(train_data)
```

In [24]:
```
# Make predictions on the testing data
predictions = pipeline.transform(test_data)
```

In [25]:
```
# Inspect results
predictions.select("prediction", "c41_idx").show(5)
```

```
+----------+-------+
|prediction|c41_idx|
+----------+-------+
|       5.0|    5.0|
|       5.0|    5.0|
|       5.0|    5.0|
|       5.0|    5.0|
|       5.0|    5.0|
+----------+-------+
only showing top 5 rows
```

In [26]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

In [29]:
```
# Select (prediction, true label) and compute test error
acc_evaluator = MulticlassClassificationEvaluator(labelCol="c41_idx",
                                                  predictionCol="prediction",
                                                  metricName="accuracy")
```

In [30]:
```
#important: need to cast to float type, and order by prediction, else it won't wo
preds_and_labels = predictions.select(['prediction','c41_idx'])\
                   .withColumn('c41_idx', col('c41_idx')\
                               .cast("float")).orderBy('prediction')
#select only prediction and label columns
preds_and_labels = preds_and_labels.select(['prediction','c41_idx'])
```

In [31]:
```
acc_evaluator.evaluate(preds_and_labels)
```

Out[31]:
```
0.9930587878174644
```

In [32]:
```
from pyspark.mllib.evaluation import MulticlassMetrics
```

In [33]:
```
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

In [34]:
```
# Confusion matrix
metrics = MulticlassMetrics(preds_and_labels.rdd.map(tuple))
```

In [35]:
```
# print(metrics.confusionMatrix().toArray())
```

In [36]:
```
import pandas as pd
pd.set_option('display.max_columns', 30)
```

In [37]:
```
matrix = pd.DataFrame(metrics.confusionMatrix().toArray())
```

In [38]:
```
matrix
```

Out[38]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0	56131.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	0.0	21492.0	102.0	0.0	4.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	0.0	0.0	19292.0	0.0	0.0	63.0	0.0	0.0	1.0	0.0	0.0	0.0	2.0	0.0	0.0	0
3	0.0	0.0	16.0	439.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4	0.0	1.0	13.0	0.0	319.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
5	0.0	0.0	31.0	0.0	0.0	220.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
6	0.0	2.0	181.0	0.0	8.0	0.0	17.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
7	0.0	0.0	184.0	0.0	0.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	162.0	0.0	0.0	0.0	0.0	0.0	0.0	0
9	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	67.0	0.0	0.0	0.0	0.0	0.0	0
10	0.0	0.0	37.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
11	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
12	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0
13	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
14	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
15	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
16	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
17	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
18	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

In [39]:
```
# With accuracy, this model is very good.
# But with confusion matrix: Some classes, which only have a few samples, have wr
```

Make new prediction

In [40]:
```
# New data
df_new = spark.read.csv('kdd_data/kddcup.testdata.unlabeled_10_percent.gz',
                        inferSchema=True,
                        header=False)
```

In [41]:
```
df_new.count()
```

Out[41]:
```
311029
```

In [42]:
```
str(df_new.columns)
```

Out[42]:
```
"['_c0', '_c1', '_c2', '_c3', '_c4', '_c5', '_c6', '_c7', '_c8', '_c9', '_c10',
'_c11', '_c12', '_c13', '_c14', '_c15', '_c16', '_c17', '_c18', '_c19', '_c20',
'_c21', '_c22', '_c23', '_c24', '_c25', '_c26', '_c27', '_c28', '_c29', '_c30',
'_c31', '_c32', '_c33', '_c34', '_c35', '_c36', '_c37', '_c38', '_c39', '_c4
0']"
```

In [43]:
```
# Make predictions on the testing data
predictions_new = pipeline.transform(df_new)
```

In [44]:
```
predictions_new.select('features', 'prediction').show()
```

```
+--------------------+----------+
|            features|prediction|
+--------------------+----------+
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[8,68,78,96,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[2,5,68,78,7...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[2,5,68,78,7...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[0,2,6,68,78...|       2.0|
|(115,[2,5,68,78,7...|       2.0|
|(115,[2,5,68,78,7...|       2.0|
|(115,[2,5,68,78,7...|       2.0|
|(115,[4,68,78,79,...|       2.0|
|(115,[4,68,78,79,...|       2.0|
+--------------------+----------+
only showing top 20 rows
```