Лабораторная работа

Использование фрэймворка Apache Spark для построения интеллектуальных систем, использующих большие данные

Выполнила: Короткова Инга Сергеевна

2020 год

Цель работы – получить навыки проектирования интеллектуальных моделей с помощью библиотеки MLlib фрэймворка Apache для задач классификации пациентов с заболеваниями печени.

Задачи:

1. Создать объект SparkSession.
2. Загрузить набор данных, содержащий результаты лабораторных исследований пациентов Indian_Liver_Patient_Dataset_ILPDh.csv, у части из которых диагностировано заболевание печени.
3. Произвести исследовательский анализ данных.
4. Разделить набор данных на обучающую и тестовую выборку.
5. Обучить модели классификации на основе алгоритмов случайный лес и логистическая регрессия.
6. Оценить эффективность моделей на тестовой выборке с помощью матрицы неточностей, критериев полноты Recall и точности Precision

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.6/dist-packages (3.0.1)
Requirement already satisfied: py4j==0.10.9 in /usr/local/lib/python3.6/dist-packages (from pyspark) (0.10.9)
```

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import LogisticRegression

```

Создание SparkSession

Создадим объект SparkSession.

spark – объект SparkSession.

```
spark = SparkSession.builder.appName('Liver Patient Dataset AnalysisApp').getOrCreate()
```

Загрузка данных

Загрузим датасет, содержащий результаты лабораторных исследований пациентов Indian_Liver_Patient_Dataset_ILPDh.csv, у части из которых диагностированоз аболевание печени.

df – объект датафрэйм, в который происходит загрузка данных из файла 'Indian_Liver_Patient_Dataset_ILPDh.csv'.

```
df = spark.read.csv('Indian_Liver_Patient_Dataset_ILPDh.csv', inferSchema=True, sep=";", header=True)
```

```
df.head()
```

```
Row(Age=65, Gender=1, Total_Bilirubin=0.7, Direct_Bilirubin=0.1, Alkaline_Phosphotase=187, Alamine_Aminotransferase=16, Aspartate_Ar
```

Произведем исследовательский анализ данных:

- получим объём исследуемых данных;
- получим число атрибутов и их типы данных;
- посмотрим распределение числа примеров классов.

```
print(f"Total rows %d and columns %d" % (df.count(), len(df.columns)))
```

```
Total rows 579 and columns 11
```

```
df.printSchema()
```

```
root
 |-- Age: integer (nullable = true)
 |-- Gender: integer (nullable = true)
 |-- Total_Bilirubin: double (nullable = true)
 |-- Direct_Bilirubin: double (nullable = true)
```

```
df.groupby('Class_ID').count().show()
```

```
+--------+-----+
|Class_ID|count|
+--------+-----+
|       1|  414|
|       2|  165|
+--------+-----+
```

Разделим набор данных на обучающую и тестовую выборку.

Для обучения модели необходимо выбрать признаки, применяемые в модели, сформировать вектор признаков features и добавить его в датафрэйм. В работе выборка делится на обучающую и тестовую в отношении 3:1 (75%: 25%).

```
df_assembler = VectorAssembler(inputCols=['Age', 'Gender', 'Total_Bilirubin', \
 'Direct_Bilirubin', 'Alkaline_Phosphotase', \
 'Alamine_Aminotransferase', 'Aspartate_Aminotransferase', \
 'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio'], outputCol="features")

df = df_assembler.transform(df)
df.printSchema()
```

```
root
 |-- Age: integer (nullable = true)
 |-- Gender: integer (nullable = true)
 |-- Total_Bilirubin: double (nullable = true)
 |-- Direct_Bilirubin: double (nullable = true)
 |-- Alkaline_Phosphotase: integer (nullable = true)
 |-- Alamine_Aminotransferase: integer (nullable = true)
 |-- Aspartate_Aminotransferase: integer (nullable = true)
 |-- Total_Protiens: double (nullable = true)
 |-- Albumin: double (nullable = true)
 |-- Albumin_and_Globulin_Ratio: double (nullable = true)
 |-- Class_ID: integer (nullable = true)
 |-- features: vector (nullable = true)
```

```
train_df,test_df=df.randomSplit([0.75,0.25])
print('Training set info:')
train_df.groupby('Class_ID').count().show()
print('Test set info:')
test_df.groupby('Class_ID').count().show()
```

```
Training set info:
+--------+-----+
|Class_ID|count|
+--------+-----+
|       1|  309|
|       2|  119|
+--------+-----+

Test set info:
+--------+-----+
|Class_ID|count|
+--------+-----+
|       1|  105|
|       2|   46|
+--------+-----+
```

Обучение моделей

Обучим модели классификации на основе алгоритмов случайный лес и логистическая регрессия.

Random Forest

```
rf_classifier=RandomForestClassifier(labelCol='Class_ID', numTrees=35).fit(train_df)
rf_predictions=rf_classifier.transform(test_df)
rf_predictions.select(['Class_ID', 'prediction']).show(30,False)
rf_predictions.printSchema()
print(rf_classifier.featureImportances)
```

```
+--------+----------+
|Class_ID|prediction|
+--------+----------+
|2       |1.0       |
|1       |1.0       |
|2       |1.0       |
|2       |2.0       |
|1       |1.0       |
|2       |1.0       |
|2       |1.0       |
|1       |1.0       |
|1       |2.0       |
|1       |1.0       |
|1       |1.0       |
|1       |1.0       |
|2       |1.0       |
|2       |1.0       |
|2       |1.0       |
|2       |2.0       |
|2       |2.0       |
|1       |1.0       |
|1       |1.0       |
|1       |2.0       |
|2       |1.0       |
|2       |1.0       |
|2       |2.0       |
|1       |1.0       |
|1       |2.0       |
|2       |1.0       |
|1       |1.0       |
|1       |1.0       |
|2       |2.0       |
+--------+----------+
only showing top 30 rows

root
 |-- Age: integer (nullable = true)
 |-- Gender: integer (nullable = true)
 |-- Total_Bilirubin: double (nullable = true)
 |-- Direct_Bilirubin: double (nullable = true)
 |-- Alkaline_Phosphotase: integer (nullable = true)
 |-- Alamine_Aminotransferase: integer (nullable = true)
 |-- Aspartate_Aminotransferase: integer (nullable = true)
 |-- Total_Protiens: double (nullable = true)
 |-- Albumin: double (nullable = true)
 |-- Albumin_and_Globulin_Ratio: double (nullable = true)
 |-- Class_ID: integer (nullable = true)
 |-- features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)

(10,[0,1,2,3,4,5,6,7,8,9],[0.1386747315496918,0.0055715843908345,0.09620664383827936,0.11052536619276124,0.1915840523012491,0.15981
```

Logistic Regression

```
log_reg=LogisticRegression(labelCol='Class_ID').fit(train_df)
lr_predictions = log_reg.transform(test_df)
lr_predictions.select(['Class_ID', 'prediction']).show(30,False)
lr_predictions.printSchema()
```

```
+--------+----------+
|Class_ID|prediction|
+--------+----------+
|2       |2.0       |
|1       |2.0       |
|2       |2.0       |
|2       |2.0       |
|1       |1.0       |
|2       |2.0       |
|2       |2.0       |
|1       |1.0       |
|1       |2.0       |
|1       |1.0       |
|1       |2.0       |
|1       |1.0       |
|2       |2.0       |
|2       |2.0       |
|2       |1.0       |
|2       |2.0       |
|2       |2.0       |
|1       |2.0       |
|1       |1.0       |
|1       |2.0       |
|2       |1.0       |
|2       |1.0       |
|2       |2.0       |
|1       |1.0       |
|1       |2.0       |
|2       |1.0       |
+--------+----------+
only showing top 30 rows
```

Оценка моделей.

Оценим эффективность моделей на тестовой выборке с помощью матрицы неточностей, критериев полноты Recall и точности Precision.

```
 |-- Aspartate_Aminotransferase: integer (nullable = true)
```

Для этого определим true_positives, true_negatives, false_positives, false_negatives – количество истинноположительных, истинно-отрицательных, ложно-положительных и ложноотрицательных случаев соответственно.

При помощи переменных recall и precision, выведем значения полноты и точности классифицирующей модели.

```
 |-- probability: vector (nullable = true)
```

```
true_positives = rf_predictions[(rf_predictions.Class_ID == 2) & (rf_predictions.prediction == 2)].count()
true_negatives = rf_predictions[(rf_predictions.Class_ID == 1) & (rf_predictions.prediction == 1)].count()
false_positives = rf_predictions[(rf_predictions.Class_ID == 1) & (rf_predictions.prediction == 2)].count()
false_negatives = rf_predictions[(rf_predictions.Class_ID == 2) & (rf_predictions.prediction == 1)].count()
print('TP = ',true_positives)
print('TN = ',true_negatives)
print('FP = ',false_positives)
print('FN = ',false_positives)
precision = float(true_positives) / (true_positives + false_positives)
recall = float(true_positives)/(true_positives + false_negatives)
print('recall = ',recall)
print('precision = ',precision)
```

```
TP =  7
TN =  95
FP =  10
FN =  10
recall =  0.15217391304347827
precision =  0.4117647058823529
```

Для логистической регрессии:

```
true_positives = lr_predictions[(lr_predictions.Class_ID == 2) & (lr_predictions.prediction == 2)].count()
true_negatives = lr_predictions[(lr_predictions.Class_ID == 1) & (lr_predictions.prediction == 1)].count()
false_positives = lr_predictions[(lr_predictions.Class_ID == 1) & (lr_predictions.prediction == 2)].count()
false_negatives = lr_predictions[(lr_predictions.Class_ID == 2) & (lr_predictions.prediction == 1)].count()
print('TP = ',true_positives)
print('TN = ',true_negatives)
print('FP = ',false_positives)
print('FN = ',false_positives)
precision = float(true_positives) / (true_positives + false_positives)
recall = float(true_positives)/(true_positives + false_negatives)
print('recall = ',recall)
print('precision = ',precision)
```

```
TP =  12
TN =  96
FP =  9
FN =  9
recall =  0.2608695652173913
precision =  0.5714285714285714
```

Заключение

В данной работе мы познакомились с библиотекой MLlib фрэймворка Apache для решения задачи классификации пациентов с заболеваниями печени.

Для этого:

- создали объект сессию SparkSession
- загрузили набор данных, содержащий результаты лабораторных исследований пациентов Indian_Liver_Patient_Dataset_ILPDh.csv, у части из которых диагностировано заболевание печени
- произвели обзорный исследовательский анализ данных
- разделили набор данных на обучающую и тестовую выборку
- обучили две модели машинного обуения для классификации пациентом: random forest и logistic regression
- оценили работу моделей на тестовой выборке при помощи матрицы неточностей, критериев полноты Recall и точности Precision.

Как итог, логистическая регрессия показала лучшие результаты, чем случайный лес.