Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра «Системи штучного інтелекту»

Розрахункова Робота

З предмету: «Видобування великих даних»

Виконав

студент групи КН-311

Ткачук Орест

Прийняла :

Шаховська Н. Б.

Львів-2021

Завдання: застосувати Mapreduce до пришвидшення роботи методів штучного інтелекту для аналізу даних.

Хід роботи

Для розрахункової роботи будемо використовувати Spark. Створюємо сесію.

```
import findspark
findspark.init("C:\\Spark\\spark-3.1.1-bin-hadoop2.7")
spark = SparkSession.builder.master("local").appName('abc').getOrCreate()
```

Читаємо оброблені дані. Замінимо колонки ‘sex’ і ‘smoker’ на бінарні значення і застосуємо ‘one-hot-encoding’ до категоріальної колонки ‘region’.

```
insurance = pd.read_csv('insurance_1.csv',engine='python', delimiter=',')
insurance['smoker'] = insurance['smoker'].replace({'yes': True, 'no': False})
insurance['sex'] = insurance['sex'].replace({'male': True, 'female': False})
insurance=pd.get_dummies(insurance)
insurance = insurance.apply(pd.to_numeric)
insurance
```

	age	sex	bmi	children	smoker	charges	region_northeast	region_northwest	region_southeast	region_southwest
0	19	False	27.900	0	True	16884.92400	0	0	0	1
1	18	True	33.770	1	False	1725.55230	0	0	1	0
2	28	True	33.000	3	False	4449.46200	0	0	1	0
3	33	True	22.705	0	False	21984.47061	0	1	0	0
4	32	True	28.880	0	False	3866.85520	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
1333	50	True	30.970	3	False	10600.54830	0	1	0	0
1334	18	False	31.920	0	False	2205.98080	1	0	0	0
1335	18	False	36.850	0	False	1629.83350	0	0	1	0
1336	21	False	25.800	0	False	2007.94500	0	0	0	1
1337	61	False	29.070	0	True	29141.36030	0	1	0	0

1338 rows × 10 columns

Тепер треба створимо датасет використовуючи датафрейм спарку, оскільки операції на Pyspark DataFrame виконуються паралельно на різних нодах в кластері.

```
insurance = spark.createDataFrame(insurance)
```

Структура датасету

```
insurance.printSchema()
```

```
root
 |-- age: long (nullable = true)
 |-- sex: boolean (nullable = true)
 |-- bmi: double (nullable = true)
 |-- children: long (nullable = true)
 |-- smoker: boolean (nullable = true)
 |-- charges: double (nullable = true)
 |-- region_northeast: long (nullable = true)
 |-- region_northwest: long (nullable = true)
 |-- region_southeast: long (nullable = true)
 |-- region_southwest: long (nullable = true)
```

Перші 20 записів

```
insurance.show()
```

```
+---+-----+------+--------+------+------------------+----------------+----------------+----------------+----------------+
|age|  sex|   bmi|children|smoker|           charges|region_northeast|region_northwest|region_southeast|region_southwest|
+---+-----+------+--------+------+------------------+----------------+----------------+----------------+----------------+
| 19|false|  27.9|       0|  true|         16884.924|               0|               0|               0|               1|
| 18| true| 33.77|       1| false|         1725.5523|               0|               0|               1|               0|
| 28| true|  33.0|       3| false|          4449.462|               0|               0|               1|               0|
| 33| true|22.705|       0| false|       21984.47061|               0|               1|               0|               0|
| 32| true| 28.88|       0| false|         3866.8552|               0|               1|               0|               0|
| 31|false| 25.74|       0| false|         3756.6216|               0|               0|               1|               0|
| 46|false| 33.44|       1| false|         8240.5896|               0|               0|               1|               0|
| 37|false| 27.74|       3| false|         7281.5056|               0|               1|               0|               0|
| 37| true| 29.83|       2| false|         6406.4107|               1|               0|               0|               0|
| 60|false| 25.84|       0| false|28923.136919999997|               0|               1|               0|               0|
| 25| true| 26.22|       0| false|         2721.3208|               1|               0|               0|               0|
| 62|false| 26.29|       0|  true|        27808.7251|               0|               0|               1|               0|
| 23| true|  34.4|       0| false|1826.8429999999998|               0|               0|               0|               1|
| 56|false| 39.82|       0| false|        11090.7178|               0|               0|               1|               0|
| 27| true| 42.13|       0|  true|        39611.7577|               0|               0|               1|               0|
| 19| true|  24.6|       1| false|          1837.237|               0|               0|               0|               1|
| 52|false| 30.78|       1| false|        10797.3362|               1|               0|               0|               0|
| 23| true|23.845|       0| false|         2395.17155|               1|               0|               0|               0|
| 56| true|  40.3|       0| false|          10602.385|               0|               0|               0|               1|
| 30| true|  35.3|       0|  true|36837.467000000004|               0|               0|               0|               1|
+---+-----+------+--------+------+------------------+----------------+----------------+----------------+----------------+
only showing top 20 rows
```

Виділення фіч і таргетної змінної

```
features = insurance.drop('charges')
assembler = VectorAssembler(inputCols=features.columns, outputCol='features')
output = assembler.transform(insurance).select('features','charges')
```

Розділення датасету на трейн і тест

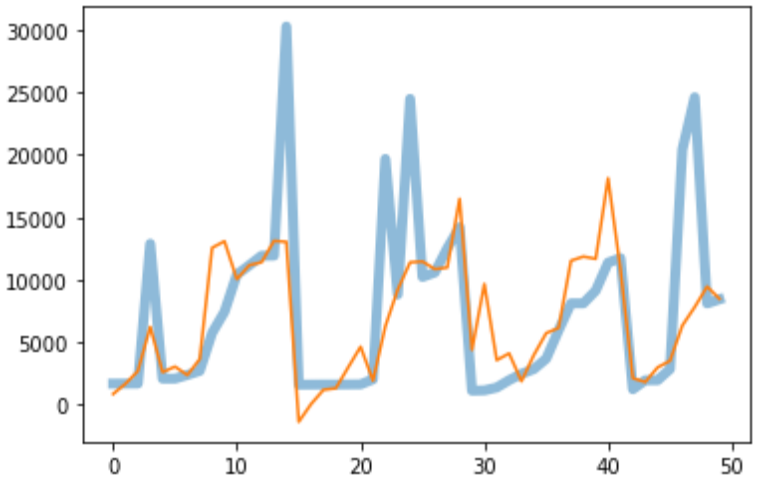
```
train, test = output.randomSplit([0.75, 0.25], seed=47)
```

Передбачення за допомогою лінійної регресії PySpark

```
lr = LinearRegression(featuresCol = 'features', labelCol='charges', maxIter=10,
regParam=0.3, elasticNetParam=0.8)
start = time.time()
lr_model = lr.fit(train)
print('fit:', time.time() - start)

start = time.time()
prediction = lr_model.transform(test)
print('pred:', time.time() - start)

plt.plot(range(50),  prediction.toPandas()['charges'][:50], linewidth=5, alpha=0.5)
plt.plot(range(50), prediction.toPandas()['prediction'][:50])
```




```
pred_eval = RegressionEvaluator(predictionCol='prediction', labelCol = 'charges',
metricName='mse')
pred_eval.evaluate(prediction)
36427107.4017943

pred_eval = RegressionEvaluator(predictionCol='prediction', labelCol = 'charges',
metricName='r2')
pred_eval.evaluate(prediction)
0.7639791067725314
```

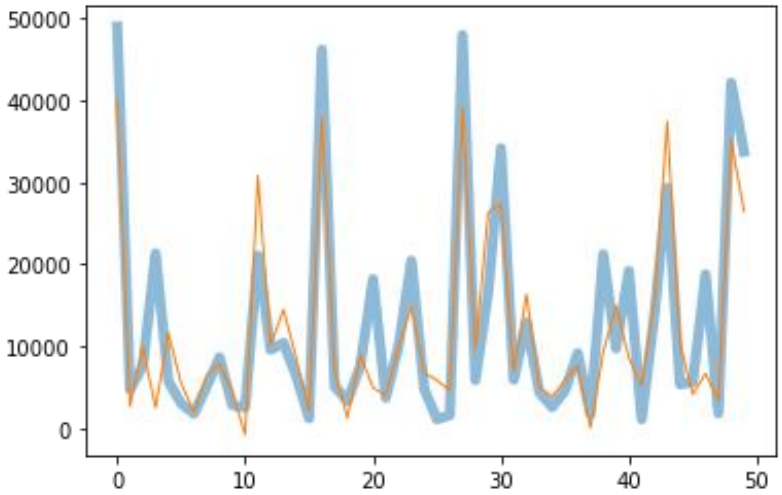
час виконання:

```
fit: 1.7345759868621826
pred: 0.01601433753967285
```

Передбачення за допомогою лінійної регресії sklearn

```
sk_insurance = insurance.toPandas()
X_train, X_test, y_train, y_test = train_test_split(sk_insurance.drop('charges', axis=1),
sk_insurance['charges'], test_size=0.25, random_state=47)
start = time.time()
sk_reg = sk_LinearRegression().fit(X_train, y_train)
print('fit:', time.time() - start)

start = time.time()
y_res = sk_reg.predict(X_test)
print('pred:', time.time() - start)
```




```
print(mean_squared_error(y_test,y_res))
print(sk_reg.score(X_test, y_test))
34120930.701114096
0.7784450657210209
```

час виконання:

```
fit: 0.1881706714630127
pred: 0.0020020008087158203
```

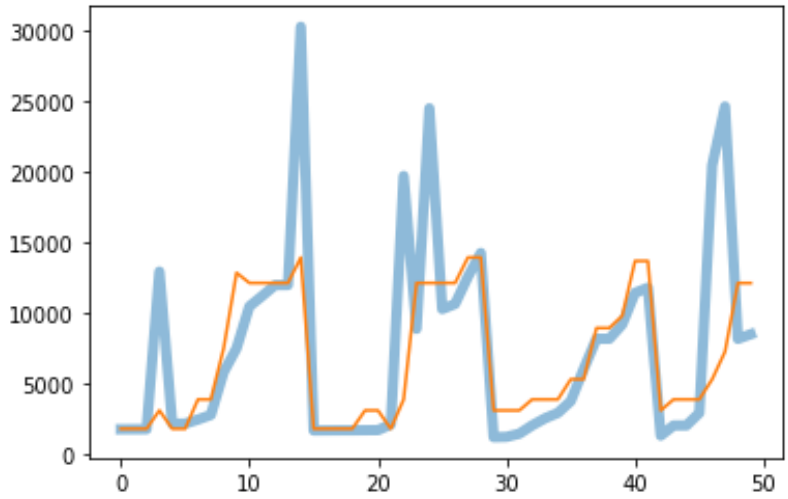
Передбачення за допомогою дерева рішень PySpark

```
dt = DecisionTreeRegressor(featuresCol='features', labelCol = 'charges', maxDepth=6,
seed=47)

start = time.time()
dt_model = dt.fit(train)
print('fit:', time.time() - start)

start = time.time()
prediction = dt_model.transform(test)
print('pred:', time.time() - start)

plt.plot(range(50),  prediction.toPandas()['charges'][:50], linewidth=5, alpha=0.5)
plt.plot(range(50), prediction.toPandas()['prediction'][:50])
```




```
pred_eval = RegressionEvaluator(predictionCol='prediction', labelCol = 'charges',
metricName='mse')
pred_eval.evaluate(prediction)
24256825.80961724

pred_eval = RegressionEvaluator(predictionCol='prediction', labelCol = 'charges',
metricName='r2')
pred_eval.evaluate(prediction)
0.8428335900707018
```

час виконання:

```
fit: 2.383165121078491
pred: 0.02402210235595703
```

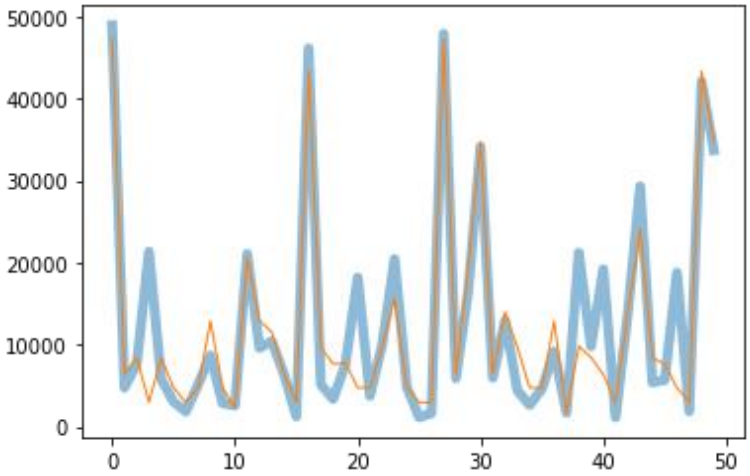
Передбачення за допомогою дерева рішень sklearn

```
sk_dt = sk_DecisionTreeRegressor(max_depth=6,  random_state=47)

start = time.time()
sk_dt.fit(X_train, y_train)
print('fit:', time.time() - start)

start = time.time()
y_res = sk_dt.predict(X_test)
print('pred:', time.time() - start)

plt.plot(range(50), y_test[:50], linewidth=5, alpha=0.5)
plt.plot(range(50), y_res[:50], linewidth=1)
```




```
print(mean_squared_error(y_test,y_res))
print(sk_dt.score(X_test, y_test))
25064922.03119789
0.837247781956063
```

час виконання:

```
fit: 0.011009454727172852
pred: 0.002002716064453125
```

Висновок: Під час даної розрахункової роботи я застосував Mapreduce за допомогою PySpark для пришвидшення роботи різних методів аналізу даних Decision Tree, та Linear Regression. З однаковими даними, Decision Tree показав кращий результат точності передбачення, ніж Linear Regression. Щодо часу виконання, то PySpark показав гірші результати ніж послідовний алгоритм. Це зумовлене тим, що PySpark змушений транслювати код в Spark і виконувати його на JVM. Тому на невеликих даних використання PySpark не є доцільним.