



Διαχείριση Μεγάλων Δεδομένων

Project 1

Όνομα: Χρήστος - Θρασύβουλος Κούλης

Αριθμός Μητρώου: en2190007

Μεταπτυχιακό: ΠΜΣ στη Μηχανική Υπολογιστών, Τηλεπικοινωνιών και Δικτύων

(MSc in Computer, Telecommunications and Network Engineering) -
Ειδίκευση Τηλεπικοινωνιών και Επεξεργασίας Σήματος

Ημερομηνία Παράδοσης : 12/6/2020

Ερώτημα 1

Εγκατέστησα το **Spark** σε λειτουργικό Linux Mint, σύμφωνα με τις οδηγίες. Με *`pip install`* εγκατέστησα το **pyspark**. Με εντολές από το command window εγκατέστησα το **scala**.

Το πρόγραμμα τρέχει κανονικά αν γράψω *spark-shell* στο command window, χωρίς προβλήματα.

```
Spark context Web UI available at http://192.168.1.2:4040
Spark context available as 'sc' (master = local[*, app id = local-1591106127107).
Spark session available as 'spark'.
Welcome to

      ____
     / __ \
    / /_/_\
   / ___/
  /_/

 version 3.0.0-preview2

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 11.0.5)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Ερώτημα 2

Έφτιαξα λογαριασμό στο Kaggle και κατέβασα το *fake_job_postings dataset*. Πρόκειται για ένα .csv αρχείο.

fake_job_postings.csv 50,1 MB Text Sat 29 Feb 2020 07:23:40 AM EET

Ερώτημα 3

Για αυτό το ερώτημα ξεκίνησα ένα spark shell, φόρτωσα το fake_job_postings.csv σε μία απλή μεταβλητή και με κάποιες απλές εντολές άντλησα πληροφορίες από αυτό.

Το dataset έγινε load με χρήση της εντολής:

```
val df =  
spark.read.format("csv").option("header", "true").load("/home/akis/Desktop/Big Data  
Projects/Project 1/fake_job_postings.csv")
```

όπου μέσα στο load είναι το directory που είχα αποθηκεύσει το αρχείο .csv.

a.) Για να βρω πόσες γραμμές έχει το αρχείο χρησιμοποίησα την εντολή:

```
df.count()
```

Που απλά μετράει πόσες γραμμές έχει το dataset. Βρήκα ότι το dataset έχει **17880 γραμμές**.

b.) Η στήλη που καθορίζει αν μία δουλειά είναι αληθινή ή ψεύτικη είναι η ['fraudulent']. Η στήλη αυτή έπαιρνε μόνο δύο τιμές, 0 και 1. Το 0 αντιστοιχούσε σε αληθινές δουλειές και το 1 σε ψεύτικες. Για να βρω πόσα job postings ήταν fake χρησιμοποίησα την εντολή:

```
val fake = df.filter($"fraudulent" === 1)
```

Που ουσιαστικά κρατάει από το αρχικό dataset μόνο όσες γραμμές ήταν fake job postings.

Βρήκα ότι οι ψεύτικες δουλειές είναι 886 γραμμές (άρα **886 fake job postings**).

c.) Ομοίως με πριν, για να βρω πόσα job postings ήταν πραγματικά χρησιμοποίησα την εντολή:

```
val fake = df.filter($"fraudulent" === 0)
```

Που ουσιαστικά κρατάει από το αρχικό dataset μόνο όσες γραμμές ήταν real job postings.

Βρήκα ότι οι αληθινές δουλειές είναι 16080 γραμμές (άρα **16080 real job postings**).

d.) Για να βρω τις κορυφαίες απαιτήσεις ασχολήθηκα με την στήλη ['required_education']. Για να βρω τις 10 κορυφαίες έγραψα αυτήν την εντολή:

```
val fakereq =  
fake.groupBy("required_education").count().orderBy(desc("count")).limit(10).show()
```

Αρχικά ομαδοποιούμε μαζί όλες τις γραμμές με παρόμοιες τιμές (*groupBy()*), και ύστερα μετράμε (*count()*) πόσες φορές εμφανίζεται η κάθε μία. Με *orderBy(desc("count"))* τα ταξινομούμε με φθίνουσα σειρά, ενώ με *limit(10)* κρατάμε μόνο τις 10 κορυφαίες. Με το *show()* τα εμφανίζουμε στην οθόνη. Με το όρισμα *show(false)* εμφανίζουμε ολόκληρο το όνομα τους. Παρακάτω είναι τα αποτελέσματα:

required_education	count
null	438
High School or eq...	164
Bachelor's Degree	96
Unspecified	60
Master's Degree	30
Some High School ...	20
Certification	18
We provide:* Comp...	5
Associate Degree	5
GoldLeaf provides...	3
BENEFITSGoldLeaf ...	3
Some College Cour...	3
Professional	3
and/or Swedish"	2
We provide:* Comp...	2
tackling operati...	2
We provide:* Our ...	1
BBA Accounting; C...	1
and having the a...	1
Passion for digit...	1

only showing top 20 rows

Παρατηρούμε ότι στα fake job postings πολλές γραμμές δεν έχουν σωστό data για αυτήν την στήλη, ενώ υπάρχουν πολλές fake και null τιμές, πράγμα που ενισχύει τις υποψίες μας ότι μία δουλειά μπορεί να είναι ψεύτικη. Το όρισμα *limit(10)* δεν έχει χρησιμοποιηθεί για τον παραπάνω πίνακα για να έχουμε πιο ολοκληρωμένη εικόνα.

e.) Ομοίως με πριν, ασχολήθηκα με την στήλη ['required_education']. Για να βρω τις 10 κορυφαίες έγραψα αυτήν την εντολή:

```
val realreq =  
real.groupBy("required_education").count().orderBy(desc("count")).limit(10).show(false)
```

Ουσιαστικά κάνουμε ακριβώς το ίδιο με πριν αλλά μόνο στο dataframe με τις αληθινές δουλειές. Τα αποτελέσματα ήταν τα παρακάτω:

required_education	count
null	7223
Bachelor's Degree	4758
High School or eq...	1799
Unspecified	1237
Master's Degree	356
Associate Degree	253
Certification	150
Some College Cour...	93
Professional	64
Vocational	47
Doctorate	22
Vocational - HS D...	9
Some High School ...	6
Vocational - Degree	4
strong negotiati...	3
We provide:* Comp...	2
Competitive baseA...	2
Medical,Dental,Vi...	1
Be part of the Ta...	1
We also party a ...	1

only showing top 20 rows

Παρατηρούμε ότι στα real job postings έχουμε πολύ πιο ξεκάθαρη εικόνα για το τι χρειαζόμαστε, παρόλα αυτά υπάρχουν ακόμα πολλές fake και null τιμές. Το όρισμα *limit(10)* δεν έχει χρησιμοποιηθεί για τον παραπάνω πίνακα για να έχουμε πιο ολοκληρωμένη εικόνα.

Όλες οι εντολές που χρησιμοποιήθηκαν για αυτό το ερώτημα βρίσκονται στο αρχείο Erotima1_3.txt. Απλά κάντε copy-paste τα περιεχόμενα του μέσα σε ένα spark-shell για να εμφανιστούν όλα τα αποτελέσματα.

Πρώτα, βρήκα ότι για τα maximum salaries στα fake job postings υπάρχει μία τιμή 180000, που είναι πάνω από 50% της αμέσως μεγαλύτερης τιμής, οπότε λογικά ίσως είναι outlier. Η τιμή αυτή αφαιρέθηκε από το dataframe.

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

  ____      __
 / ___ |____|  |  __ _ _ __| | | |
 \___ \|___ \|__| | | | |___| | | |
  ___) |___) |___| | | | |___| | | |
 |____|____|____|_|_|_|_____|_|_|_|

 version 3.0.0-preview2

Using Python version 3.6.9 (default, Nov  7 2019 10:44:02)
SparkSession available as 'spark'.
>>> 
```

Για τα minimum salaries των real job postings υπάρχουν πιο προφανή outlier. Είδα ότι υπάρχουν αρκετές τιμές σχεδόν μηδενικές, κάτι που δεν βγάζει νόημα, οπότε αφαίρεσα όλα τα job postings με minimum μισθό λιγότερο των 1000 ευρώ. Ακόμα, υπάρχουν 2 τιμές μισθών άνω των 100 εκατομμυρίων που προφανώς δεν βγάζουν νόημα. Οι τιμές λοιπόν με minimum μισθό άνω των 600000 ευρώ αφαιρέθηκαν.

Για το f.), τα νέα αποτελέσματα είναι:

$$average = 54180.55$$

$$standard\ deviation = 27167.15$$

Ενώ για το g.):

$$median = 40000$$

Βλέπουμε ότι δεν άλλαξε, κάτι που είναι λογικό καθώς ο median δεν εξαρτάται από outliers. Ο τρόπος που υπολογίστηκαν τα παραπάνω βρίσκονται στον κώδικα.

i.) Για να βρω τα bigram και τα trigram συγκέντρωσα όλα τα description σε list (και ύστερα string) μεταβλητές και χρησιμοποίησα τις βιβλιοθήκες *re*, *ngrams* και *collections* της *pythhon* για να τα απαριθμίσω. Τα αποτελέσματα ήταν τα παρακάτω:

Για τις ψεύτικες δουλειές:

Bigram	Occurences	Trigram	Occurences
of , the	462	are , looking , for	184
we , are	412	we , are , looking	145
in , the	397	oil , and , gas	128
looking , for	344	looking , for , a	124
to , the	288	and , gas , industry	98
for , the	285	we , are , seeking	91
to , work	276	be , responsible , for	85
in , a	237	be , able , to	74
is , a	235	as , well , as	70
are , looking	218	we , are , a	64

Για τις αληθινές δουλειές:

Bigram	Occurences	Trigram	Occurences
of , the	9722	you , will , be	2923
in , the	8611	are , looking , for	2920
will , be	8352	we , are , looking	2678
you , will	7724	looking , for , a	2576
we , are	7140	as , well , as	2160
looking , for	6404	be , responsible , for	1636
in , a	5184	to , join , our	1584
is , a	5010	be , able , to	1530
to , the	4811	will , be , responsible	1376
for , the	4794	this , is , a	1365

*Ο κώδικας βρίσκεται στο αρχείο Erotima1_4.py. Πρόκειται για ένα απλό python
αρχείο.*

Ερώτημα 5

Για το ερώτημα αυτό χρησιμοποιήθηκε η python, όχι όμως σε περιβάλλον pyspark.

j.) Αρχικά θα χρησιμοποιήσω μόνο την στήλη ['telecommuting'] ως feature σε έναν naive bayes classifier. Τα αποτελέσματα εδώ όπως θα δούμε και στην παρακάτω εικόνα δεν είναι καθόλου ικανοποιητικά, με *precision*, *recall* και *f1_score* κάτω του 10%. Αντίθετα, όταν ενσωματώσουμε στα feature list και τις στήλες ['has_company_logo'] και ['has_questions'], το ποσοστό επιτυχίας αυξάνεται ραγδαία (πάνω του 90%), όταν χρησιμοποιήσουμε k-Nearest Neighbors classifier με N=10 γείτονες ή SVM με sigmoid kernel. Για το SVM το precision είναι μικρότερο, καθώς όπως είναι γνωστό τα sigmoid kernel δεν βγάζουν τόσο καλά αποτελέσματα.

Τα *precision*, *recall* και *f1-score* που υπολόγισα για τα 3 μοντέλα είναι τα παρακάτω:

```
PS C:\Users\AK> python -u "c:\Users\AK\Desktop\METATPYXIAKO\Μεταπτυχιακό ΕΠΠΑ Τηλεπικοινωνίες\2ο Εξάμηνο\Διαχείριση Μεγάλων Δεδομένων\Big Data Projects\Project 1\για αναφορά\1-5a.py"

Naive bayes estimator Performance:

Precision = 0.08344198174706649
Recall = 0.07390300230946882
F1 score = 0.07838334353949786

-----

kNN nearest neighbors estimator Performance (10 neighbors):

Precision = 0.9501118568232663
Recall = 0.9501118568232663
F1 score = 0.9250859093560655

-----

SVM linear kernel estimator performance:

Precision = 0.9071193285300408
Recall = 0.9409395973154362
F1 score = 0.9226417381929095
```

Ο κώδικας βρίσκεται στο αρχείο 1-5a.py Πρόκειται για ένα απλό python αρχείο.