Spark Project Report 2

Shun Zhang
School of Data Science
Fudan University
Shanghai, China
15300180012@fudan.edu.cn

1 Introduction

Recently, there have been serious data breaches in Turkey. The personal information of about 50 million Turkish citizens has been hacked onto the Internet. The information includes names, ID numbers, birth dates, parents' names, and residences. Roughly speaking, this will be the largest national database ever leaked.

On April 3, 2016, a series of sensitive information, including names, ID numbers, parents' names, addresses, etc., was hacked and packaged under an IP address in Finland. The scale of these data is 6.6G and the compressed version is 1.6G. The Associated Press randomly verified the information of 10 non-public figures. Among the 10 people, 8 had the right number, and the accuracy rate was quite high.

According to data of 2014, Turkey's population is close to 76 million. This means that the number of people affected by the leak will exceed 70% of the country's total population. Since the identity information of the Turkish identity card is tied with a number of government projects such as taxation, voting, social insurance, medical insurance, and military recruitment, risks such as "identity theif" and "fraudulent crime" may arise.

However, as for us, undergraduates, it's interesting to do some analysis about these data with no harm.

Remark:

- *All the related codes are attached in the same zip file. The file name for .py denotes which problem it solves.*
- *The word 'we' within this report actually means myself, while it's better in writing.*
- *Some of the experiments are just run on my laptop with a rather small dataset, which will be clearly claimed.*
- *Some of the experiments haven't been run on test set, because of the limited resources. However, most of the models are just NaiveBayes, which do not need any parameter tuning work, so the performance on validation set will be enough. Other model like Logistic Regression has been run on test set, see Table 6.*

2 Problem N6

Here our goal is to find the top-5 popular first names of males and females respectively. And our solution only requires one shuffle dependency: *reduceByKey* to obtain the frequency of the names.

results The results is shown in Table 1

Table 1: Top-5 popular first names of males and females

Name rank	Males	Females
1	MEHMET(821178)	FATMA(808397)
2	MUSTAFA(628867)	AYSE(625000)
3	AHMET(503835)	EMINE(529607)
4	ALI(463808)	HATICE(461773)
5	HUSEYIN(365188)	ZEYNEP(221004)

3 Problem N7

Here our goal is to find the top-3 names for every top-10 city(address city) in population. Our pipeline goes like the following:

1. *map*: record ⇒ ((address_city, first_name), 1)
2. *reduceByKey*: ⇒ ((address_city, first_name), N)
3. *map*: ((address_city, first_name), N) ⇒ (address_city, (first_name, N))
4. *groupByKey*: ⇒ (address_city, [(first_name1, N1), (first_name2, N2), $\cdots$])
5. *takeOrdered*: take the top-10 cities in population by sum up all the name frequencies within one city

Finally, we have all the name-frequency pairs within the top-10 cities in population and we only need to take the top-3 first names for each of them. The results are shown in Table 2

Table 2: Top-3 first names for each of top-10 cities in population

City	First popular name	Second popular name	Third popular name
ISTANBUL	FATMA(108179)	MEHMET(98425)	MUSTAFA(88232)
ANKARA	FATMA(40781)	MEHMET(35748)	MUSTAFA(35473)
IZMIR	MEHMET(41396)	FATMA(35991)	MUSTAFA(32029)
BURSA	FATMA(28429)	MEHMET(23435)	AYSE(23257)
AYDIN	MEHMET(32003)	FATMA(24490)	AYSE(21322)
ADANA	MEHMET(26165)	FATMA(22225)	MUSTAFA(18604)
KONYA	MEHMET(30796)	AYSE(30040)	MUSTAFA(28603)
ANTALYA	MEHMET(26998)	FATMA(25641)	AYSE(24147)
MERSIN	MEHMET(22217)	FATMA(21569)	AYSE(18944)
KOCAELI	FATMA(14936)	AYSE(11999)	MEHMET(11536)

4 Problem H1

4.1 Analysis and insight

Here, our goal is to build a city prediction model, which will predict a citizen's **city** based on all the other information. Note that, here, we have three 'different '**cities** within one record, *birth_city*, *id_registration_city* and *address_city*. We will discuss the three cases respectively.

- **id_registration_city** and **address_city**
  The solution for these two cases is quite the same and tricky, because the other information includes **id_registration_district** and **address_district**, which implicitly contains the information of id_registration_city and address_city, respectively. For example, if a Chinese citizen lives in *Yangpu* district, then it is quite sure that he now lives in *Shanghai*, unless somewhere else exists a district of the same name, which rarely happens. This tricky relation is further confirmed by the results in Table 3, with NaiveBayes model.

  where Model 1 denotes the one predicts id_registration_city by id_registration_district, Model 2 denotes the one predicts address_city by address_district.

Table 3: Results for Naive-Bayes models

Model	train (Top-1)	val (Top-1)	train (Top-5)	val (Top-5)	test (Top-1)
1	0.9989	0.9989	1.0	0.9989	0.9315
2	0.9969	0.9969	1.0	0.9969	0.8972

- **birth_city**
  In order to predict z citizen's birth city, we could model the national migration trend. To find whether citizen from city A is more likely to settle down in city B. Hence, we could predict A given B. So, we choose the *id_registration_district* and *address_district* as our features and they are projected into a two-dimension vector, which assures that $x[0] == y[0]$ if x and y registered in the same district, $x[1] == y[1]$ if x and y now lives in the same district. Again, we build a NaiveBayes model and the results are shown in Table 4.

Table 4: Results for Naive-Bayes models(on small dataset)

Accuracy type	train	val
Top-1	0.0537	0.0542
Top-5	-	0.1160

The top-5 accuracy on training set is missing while we can see that the top-5 accuracy on validation set grows a little.

5 Problem H2

5.1 Analysis and insight

Here, our goal is to build a gender prediction model, which will predict a citizen's **gender** based on all the other information. An inspiration is that from Problem N6, Table 1, we can see that the first names of males and females are very different, which is also the case here in our country, where you will have a great confidence in identifying one's gender by one's first name.

5.2 Model and results

First, for a classic binary classification, an obvious way is to build a Logistic Regression model. The only feature is *first name*, which is encoded into one-hot vector within our pipeline. Regularization is not included here, which will be discussed later.

On the other hand, a naive but powerful model like NaiveBayes is also implemented with the same one-hot feature. The results of the two models is shown in Table 5.

Table 5: Results for LR and NB models

Model	training accuracy	validation accuracy	test accuracy
LR	0.9645	0.9633	0.9166
NB	0.9866	0.9819	0.9387

Remark: An interesting thing is that, with one-hot feature encoding, Logistic Regression will converge to the same as NaiveBayes. The reason is that with one-hot encoding, a certain dimension is correlated to a certain first name and because of 'one-hot', we have

$$logit(p_i) = \beta_i x_i = \beta_i$$

where $x_i = 1$, β_i is the coefficient of x_i, p_i is the predicted probability and function $logit(x) = log(\frac{x}{1-x})$. Furthermore, the $logit$ term could be interpreted as the likelihood of the prediction to be positive. Meanwhile, for NaiveBayes, we have

$$P(positive|x_i) \propto P(positive, x_i)$$

where $P(positive|x_i)$ is the probability of being positive given x_i and $P(positive, x_i)$ is the proportion of $(positive, x_i)$ among all the $(label_j, x_j)$ pairs.

By comparing the two equations, it's easy to find that in this situation, the converged logistic regression with converged coefficients β_i^* will have the property that

$$\forall i, j, P(positive, x_i) > P(positive, x_j) \;\Rightarrow\; \beta_i^* > \beta_j^*$$

which means that the prediction of a converged Logistic Regression and a NaiveBayes should be the same.

And from this, we can see why regularization for logistic regression is not included.

5.3 Another interesting thing

The two models mentioned above is implemented with Dataframe-based APIs in *ml* package. However, at the very first time, I used the RDD-based APIs in *mllib* package, where I used a feature, different from one-hot, encoding called *HashingTF*, which actually encodes a string into a alphabet frequency vector. For example, name "ABAC" is the same as "BAAC" with *HashingTF*. Amazingly, with this encoding, we also obtain high accuracy as shown in Table 6.

Table 6: Results for LR with different regularization coefficients (with HashingTF in mllib)

Model	training accuracy	validation accuracy	test accuracy
LR(α = 0.01)	0.9639	0.9625	0.9161
LR(α = 0.1)	0.9642	0.9629	0.9164
LR(α = 1)	**0.9645**	**0.9633**	**0.9167**
LR(α = 10)	0.9641	0.9629	0.9163

From the table, we could see that regularization do helps a little. And why regularization is included here is that, *HashingTF* encoding only requires dimension of 26, which is much smaller than that of one-hot encoding.

6 Problem H3

6.1 Analysis and insight

Here our goal is to find the latent pattern within the national identification number of a Turkish citizen. For lack of calculation resources, I only manage to predict the last number, the last two numbers and the last three numbers of the national ID. Apparently, predict n numbers means that there are 10^n labels to deal with, which grows exponentially fast.

6.2 Model and results

Considering the possible information that determines a national ID, I take the *gender*, *birth date*, and *id registration district* as the features. Considering one-hot encoding will have the problem of dimension explosion, so the three features are projected into a three dimension vector for model training, which assures that $x[0] == y[0]$ if x and y have the same gender, $x[1] == y[1]$ if x and y were born on the same day, $x[2] == y[2]$ if x and y registered in the same district.

Till now, I only manage to run the model locally on two partitions of the *val_set*, with *randomSplit* to train and val. The results are shown in Table 7.

Table 7: Results for NB models predicting ID(on small dataset)

Prediction pattern	training accuracy	validation accuracy
*X	0.2012	0.1996
*XX	0.0201	0.0198
*XXX	0.0021	0.0020

7 Problem H4

7.1 Analysis and insight

Similarly, H4 also has the problem of dimension explosion because there are more than 10000 last names, which means there are more than 10000 labels and requires a quite large driver memory to build any ML model. So, here we simply select the data with Top-20 last names. Hence, our problem is reduced to a 20-label classification task. Speaking of feature, we only choose the *address district*, hoping that people with the same last name tend to live together, like a family.

7.2 Model and results

Again, we build a NaiveBayes model with feature of one-hot encoding. The top-1 and top-5 accuracy are shown in Table 8.

Table 8: Results for NB models predicting last name(on small dataset)

Accuracy type	training accuracy	validation accuracy
Top-1	0.1296	0.1294
Top-5	0.1375	0.1331