RESEARCH

Introduction of Profile Areas of Data Science :Project 4

Suresh Kumar Choudhary, Aman Jain and Frenny Macwan

Full list of author information is available at the end of the article

Abstract

Goal of the Project:
Task 1 : To gather statistical information about the data along with preprocessing the data like imputation.
Task 2 : Using spark to develop evaluate and compare two predictors to predict the probability of a stroke happening to a patient, for both balanced and unbalanced dataset.

Main Result of the Project: Data exploration and comparing the accuracy of 2 algorithms for balanced and unbalanced data set using Spark.

Personal Key Leanings: We have done exploratory data analysis and implemented machine learning models using spark instead of using traditional techniques.

Estimate working hours: 8

Project Evaluation: 1

Number of Words: 1245

Keywords: Data Cleaning; Spark; SQL;

Scientific Background

Task 1 : In real world, the datasets presented are quite unbalanced and are huge. Data exploration helps create a more straightforward view of datasets rather than pouring over thousands of figures in unstructured data. Understanding and interpreting data from large data sets can be very challenging. Using different data exploratory data analysis methods and visualization techniques will ensure you have a richer understanding of your data.

Task 2 : A stroke occurs when the blood supply to part of your brain is interrupted or reduced, preventing brain tissue from getting oxygen and nutrients. Brain cells begin to die in minutes. A stroke is a medical emergency, and prompt treatment is crucial. Early prediction of a stroke will be extremely useful in medical applications by using proper machine learning techniques with accuracy at least more than 0.9.

Data Analysis and Cleansing task

Dataset [1] contains 43400 patient information which includes details such as age and gender, along with several health parameters (e.g. hypertension, body mass index) and lifestyle related variables (e.g. smoking status, occupation type). The data structure of the data can be found in figure 7.

The values of categorical datas for few features are as follows, which can also be found in figure 8 :

- **Gender** - Male, Female
- **Ever Married** - No, Yes
- **Work Type** - Children, Private, Never Worked, Self Employed, Gov Job
- **Residence Type** - Rural, Urban
- **Smoking Status** - Never Smoked, Formely Smoked and Smokes.
- **Stroke** - 0,1

We performed data analysis, cleansing and imputation task in python using pyspark and pyspark.sql functionalities. We followed the various steps to process and clean the data as:

1 We firstly checked the missing values in the dataset and found two columns with the missing values names as 'bmi' and 'smoking_status'. So we filled all the missing values of column 'smoking_status' with value as 'No Info' and column 'bmi' with mean of it. We tried to fill the missing values of column 'bmi' with individual class mean but there was no improvement in model accuracy so we used the whole column mean for it.

2 All columns had been typecasted as per their values explicitly like big int to int , double to float etc.

3 As a part of data analysis, We checked the distribution of all categorical columns with the help of pyspark statistics crosstab function and pyspark.sql against the 'stroke' column. However we didn't get anyinsights from it that can improve the model accuracy.

4 We calculated the correlation among the numerical columns (features) and found that all columns are important for the model as correlation value was very less.

5 We found 5 columns with categorical values with the help of column datatype 'object', which are 'Gender', 'Ever_Married','work_type','residence_type' and 'smoking_status'

6 After it we converted all categorical columns into vectors with the help of pyspark's StringIndexer and OneHotEncoderEstimator class functionalities.

7 With the help of pyspark's VectorAssembler class, we prepared a feature set for individual sample including all samples by combining these vectors and numerical columns

8 finally we checked for duplicates in data, which did not occur

1 Result

Task 1 : As we started seeing target distribution, This is an Imbalanced dataset, where the number of observations belonging to one class is significantly lower than those belonging to the other classes. In this case, the predictive model could be biased and inaccurate. We checked various features having an impact : influence of work type on getting stroke-it is mostly happening to private or self-employed person(Figure 1). Then we checked on gender feature - here 1.68% female and almost 2% male had stroke. Then we checked on age feature - here we saw that 91.5% stroke had occured for person who are more than 50 years old. (Figure 2)

As a part of task1, we have found missing values, imputed it, cleaned data, computed

correlation and converted all categorical columns into vectors(numerical columns) for model building purpose.

Figure 1 Based on work type

```
# sql query to find the number of people in specific work_type who have had stroke and not
spark.sql("SELECT work_type, COUNT(work_type) as work_type_count FROM table WHERE stroke == 1 GROUP BY work_type ORDER BY COU
spark.sql("SELECT work_type, COUNT(work_type) as work_type_count FROM table WHERE stroke == 0 GROUP BY work_type ORDER BY COU
```

```
+-------------+---------------+
|    work_type|work_type_count|
+-------------+---------------+
|      Private|            441|
|Self-employed|            251|
|     Govt_job|             89|
|     children|              2|
+-------------+---------------+

+-------------+---------------+
|    work_type|work_type_count|
+-------------+---------------+
|      Private|          24393|
|Self-employed|           6542|
|     children|           6154|
|     Govt_job|           5351|
| Never_worked|            177|
+-------------+---------------+
```

Figure 2 Based on Gender and Age

```
spark.sql("SELECT gender, COUNT(gender) as gender_count, COUNT(gender)*100/(SELECT COUNT(gender) FROM table WHERE gender ==
spark.sql("SELECT gender, COUNT(gender) as gender_count, COUNT(gender)*100/(SELECT COUNT(gender) FROM table WHERE gender ==
```

```
+------+------------+------------------+
|gender|gender_count|        percentage|
+------+------------+------------------+
|  Male|         352|1.9860076732114647|
+------+------------+------------------+

+------+------------+------------------+
|gender|gender_count|        percentage|
+------+------------+------------------+
|Female|         431|1.6793298266121177|
+------+------------+------------------+
```

```
spark.sql("SELECT COUNT(age)*100/(SELECT COUNT(age) FROM table WHERE stroke ==1) as percentage FROM table WHERE stroke == 1 A
```

```
+-----------------+
|       percentage|
+-----------------+
|91.57088122605364|
+-----------------+
```

Task 2 : Using spark, we developed, trained and compared the prediction for two different algorithms (Logistic and Random Forest) for both unbalanced and balanced data. We were given the unbalanced data and using oversampling techniques we extended the minor class and made the complete dataset as balanced dataset. We splitted the dataset into train set(70%) and test set(30%). So , we calculated the results with both algorithms as follows:

Model Training with Unbalanced Dataset:
We ran two alrogithms (Random Forest and Logistic Regression) for the unbalanced data set and tested on test dataset and got the Accuracy, F1-score and AUC for both algorithm, which can be found in Table 1. Recall, precision and confusion matrix for both models on test set can be found in figure 3 and figure 4.

Model Training with Balanced Dataset:
For balancing the training data set, first we calculated the ratio of major vs minor labels and we found the ratio of 54:1. We oversampled the minority labeled rows by

duplicating the rows using the following code :
oversampled-df = minor-df.withColumn("dummy", explode(array([lit(x) for x in a]))).drop('dummy')
After this we got a perfect balanced training data set and performed two algorithms on the balanced data set. For Logistic and Random Forest algorithms we calculated the Accuracy, F1 Score and AUC on test set, which can be found in table 1. Recall, precision and confusion matrix for both models on test set can be found in figure 5 and figure 6.

Table 1 Result Summary for unbalanced and balacned dataset with logistic regression and random forest(RF)

Scores	Unbalanced-RF	Unbalanced-Logistic	Balanced-RF	Balanced-Logistic
Accuracy	98.22	98.22	69.06	73.96
F1 Score	97.34	97.34	80.35	83.45
AUC	0.83	0.85	0.84	0.85

The confusion matrix for the 2 Classifiers (For Unbalanced and Balanced Data Set) :

Discussion-1

If we see the confusion matrix for the models trained on unbalanced dataset, we find that both models have prediction entries for class 1 (a stroke) are zero, that means model is biased towards class 0(not a stroke), which is not fulfilling our goal. After balancing the unbalanced train data set and retraining the same algorithms again we found a drastic decrease in the accuracy and F1 Score. Now we can see that precision and recall of class 1 have increased compare to unbalanced data model. So, we can conclude that after balancing the data we got better results compare to unbalanced one and trained model on balanced data is unbiased and stable.

Discussion-2

Current project dataset is having limited features and not covering enough information to predict the percentage of patient getting stroke or not. Even though we tried to train a model in the most optimal way after balancing the data, the results were not as we expected, hence we would say it is not a data science project. However, with more essential features and balanced set of data, this can be taken as a typical data science project, on which the medical doctors can rely upon.

Appendix

Code -

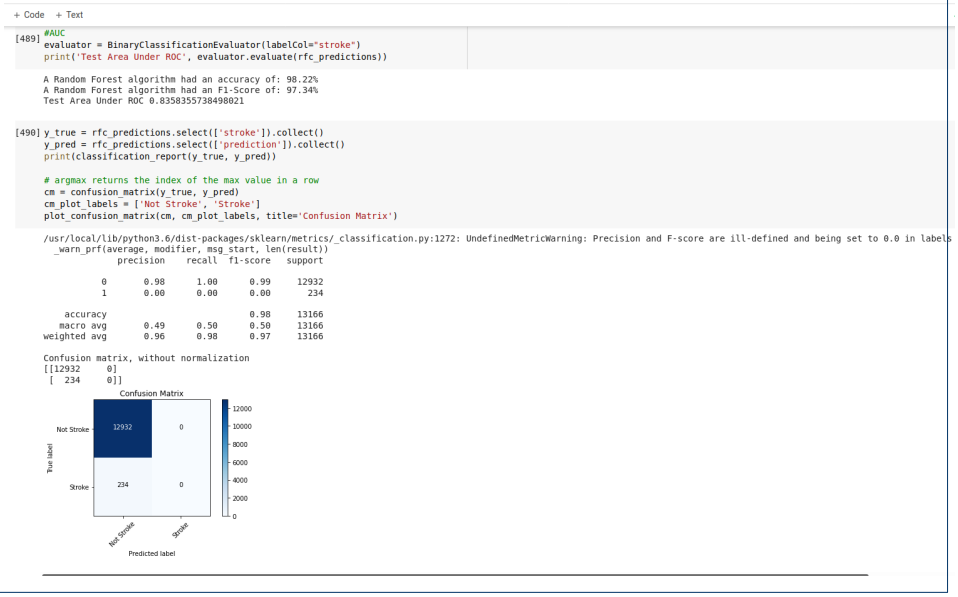
- Aman - Training algorithms with unbalanced datasets.
- Suresh - Spark Module for Task 2 and Training algorithms with balanced datasets.
- Frenny - Gathered statistical information, correlation between attributes and data imputation

Report -

- Aman - Abstract, Scientific Background
- Suresh - Results, Data Analysis and Cleaning Task, Discussion 1
- Frenny - Goal, Discussion 2

we have written the code with the help of the kaggle site [2]

Figure 3 Random Forest Model with Unbalanced Dataset



Author details

References

1. Data. https://github.com/aman1002/McKinseyOnlineHackathon-Healthcare-
2. Kaggle Pyspark. https://www.kaggle.com/njalan/healthcare-dataset-stroke-data-pyspark

Figure 4 Logistic Regression Model with Unbalanced Dataset

```
+ Code  + Text

[492] evaluator = BinaryClassificationEvaluator(labelCol="stroke")
      print('Test Area Under ROC', evaluator.evaluate(lg_predictions))

      A Logistic Regression algorithm had an accuracy of: 98.22%
      A logistic algorithm had an F1-Score of: 97.34%
      Test Area Under ROC 0.8557712135271681

      y_true = lg_predictions.select(['stroke']).collect()
      y_pred = lg_predictions.select(['prediction']).collect()
      print(classification_report(y_true, y_pred))

      # argmax returns the index of the max value in a row
      cm = confusion_matrix(y_true, y_pred)
      cm_plot_labels = ['Not Stroke', 'Stroke']
      plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')

      /usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetric
        _warn_prf(average, modifier, msg_start, len(result))
                    precision    recall  f1-score   support

                 0       0.98      1.00      0.99     12932
                 1       0.00      0.00      0.00       234

          accuracy                           0.98     13166
         macro avg       0.49      0.50      0.50     13166
      weighted avg       0.96      0.98      0.97     13166

      Confusion matrix, without normalization
      [[12932     0]
       [  234     0]]
```

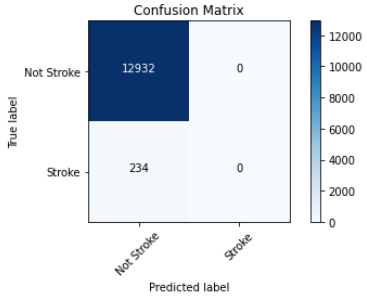
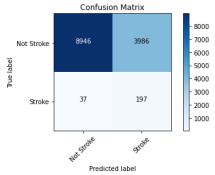


Figure 5 Random Forest Model with balanced Dataset

```
+ Code  + Text

[497] #AUC
      evaluator = BinaryClassificationEvaluator(labelCol="stroke")
      print('Test Area Under ROC', evaluator.evaluate(rfc_predictions))

      A Random Forest algorithm had an accuracy of: 69.44%
      A Random Forest algorithm had an F1-Score of: 80.35%
      Test Area Under ROC 0.8411891855094761

[498] y_true = rfc_predictions.select(['stroke']).collect()
      y_pred = rfc_predictions.select(['prediction']).collect()
      print(classification_report(y_true, y_pred))

      # argmax returns the index of the max value in a row
      cm = confusion_matrix(y_true, y_pred)
      cm_plot_labels = ['Not Stroke', 'Stroke']
      plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')

                    precision    recall  f1-score   support

                 0       1.00      0.69      0.82     12932
                 1       0.05      0.84      0.09       234

          accuracy                           0.69     13166
         macro avg       0.52      0.77      0.45     13166
      weighted avg       0.98      0.69      0.80     13166

      Confusion matrix, without normalization
      [[8946 3986]
       [  37  197]]
```




```
[499] #Logistic Regression
```

Figure 6 Logistic RegressionModel with balanced Dataset

A Random Forest algorithm had an F1-Score of: 83.45%
Test Area Under ROC 0.8554549636362195

```
y_true = lg_predictions.select(['stroke']).collect()
y_pred = lg_predictions.select(['prediction']).collect()
print(classification_report(y_true, y_pred))

# argmax returns the index of the max value in a row
cm = confusion_matrix(y_true, y_pred)
cm_plot_labels = ['Not Stroke', 'Stroke']
plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')
```

```
              precision    recall  f1-score   support

           0       1.00      0.74      0.85     12932
           1       0.05      0.81      0.10       234

    accuracy                           0.74     13166
   macro avg       0.52      0.77      0.47     13166
weighted avg       0.98      0.74      0.83     13166

Confusion matrix, without normalization
[[9549 3383]
 [  45  189]]
```

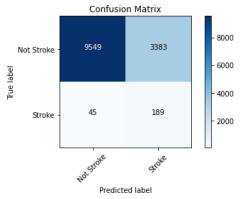


Figure 7 Dataset Details

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	30669	Male	3.0	0	0	No	children	Rural	95.12	18.0	No Info	0
1	30468	Male	58.0	1	0	Yes	Private	Urban	87.96	39.2	never smoked	0
2	16523	Female	8.0	0	0	No	Private	Urban	110.89	17.6	No Info	0
3	56543	Female	70.0	0	0	Yes	Private	Rural	69.04	35.9	formerly smoked	0
4	46136	Male	14.0	0	0	No	Never_worked	Rural	161.28	19.1	No Info	0

Figure 8 Dataset Feature Categories

```
print(train.gender.unique())
print(train.ever_married.unique())
print(train.work_type.unique())
print(train.Residence_type.unique())
print(train.smoking_status.unique())
print(train.stroke.unique())
```

```
['Male' 'Female' 'Other']
['No' 'Yes']
['children' 'Private' 'Never_worked' 'Self-employed' 'Govt_job']
['Rural' 'Urban']
['No Info' 'never smoked' 'formerly smoked' 'smokes']
[0 1]
```