Part 4 (1) – Cross Validation

In Part 4, we are required to use the 14 features obtained in Question-3 and to perform 10-fold Cross Validation across data.

The features are organized in the form of (features, predicant) pairs for each window. The feature data is split into 10 parts, such that 90% of our data will be used for fitting our model and 10% of the data will be used for testing the model.   
The process mentioned above, is performed 10 times on the feature data for each of our hastags. To evaluate the performance of the model, we use Prediction error for every fold.

Prediction error is calculated as = |N*predicted* – N*real*|

The accuracy results obtained across various hash-tags and over every fold given below,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fold No | #gopatriots | #gohawks | #nfl | #patriots | #sb49 | #superbowl |
| (1) | 5.497 | 3.624 | 5.8562 | 20.235 | 55.370 | 26.962 |
| (2) | 5.811 | 3.649 | 6.581 | 20.355 | 48.427 | 27.006 |
| (3) | 8.896 | 5.692 | 6.732 | 21.208 | 97.462 | 26.949 |
| (4) | 90.584 | 9.758 | 41.497 | 32.991 | 53.527 | 30.424 |
| (5) | 16.709 | 195.137 | 275.896 | 137.657 | 151.170 | 63.362 |
| (6) | 15.890 | 669.995 | 161.242 | 1033.888 | 300.923 | 1004.981 |
| (7) | 13.719 | 143.188 | 159.437 | 416.451 | 1726.575 | 231.682 |
| (8) | 12.524 | 151.169 | 349.7592 | 362.720 | 9300.742 | 904.430 |
| (9) | 289.840 | 828.921 | 780.542 | 3553.646 | 938.063 | 11322.880 |
| (10) | 5.756 | 8.919 | 300.279 | 108.647 | 278.735 | 708.964 |
| Average Error | 46.523 | 202.005 | 208.782 | 570.780 | 1295.099 | 1434.764 |

Figure: Average Error of 10 Fold Cross Validation

Observation:

* We can see that there is a relationship between the number of tweets for a hash-tag and the average error of cross validation. Greater the number of tweets leads to a higher absolute average error for the hash-tag.
* In particular, it is seen that for each hash-tag the error of one of the cross-validation fold is too high due to the uneven distribution of the data-set. A fold might consider a split wherein the test-data has all high values for the class (tweets during the time of the SuperBowl) and training-data has all low values for the class (tweets before and after the SuperBowl), hence producing a high error value for that fold.

Part 4 (2) - Cross Validation with Time Periods

The second part of Question-4 deals with analysis of regression models created for different time-periods during the SuperBowl. Three different time-periods were considered to create the regression models,

1. Before Feb. 1, 8:00 a.m. [when the hashtags haven’t become very active]
2. Between Feb. 1, 8:00 a.m. and 8:00 p.m. [active period]
3. After Feb. 1, 8:00 p.m. [after they pass their high-activity time]

Each tweet was segregated based on the time it was posted and split into windows of one-hour. The models were tested using 10-fold Cross Validation and the average errors for all folds obtained were as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| HashTag | Period 1 | Period 2 | Period 3 |
| #gohawks | 200.038 | 5391.083 | 3619.449 |
| #gopatriots | 15.037 | 5511.565 | 3.407 |
| #nfl | 129.083 | 6274.101 | 320.641 |
| #patriots | 193.210 | 35029.398 | 119.486 |
| #sb49 | 99.697 | 89845.155 | 233.074 |
| #superbowl | 242.084 | 894816.135 | 456.501 |

Figure: Average Error of 10 Fold Cross Validation for each Time-Period

Observation:

* The error seems to be extremely high for period 2. The reason can be that it is extremely difficult to achieve high accuracy using 12 training points. To deal with this problem we could use sliding windows to increase the number of data points

Part 5 - Testing Data

In this part, we test the models trained by us in part 4 and try to predict the values for the next hour.

The testing data was downloaded and for each file in the testing data features were collected using methods employed in the previous questions. There were 10 files in all, each of them corresponding to one of the three time periods. However, unlike before, the files had a mixture of all hashtags. But the models we had trained earlier were specific to a specific hashtag. So, we found the most dominant hashtag in each of the ten files. The dominant hashtags were:

|  |  |  |  |
| --- | --- | --- | --- |
| Test File | | Model Used | Dominant HashTag |
| Sample1\_period1 | Superbowl model for period1 | | #superbowl |
| Sample2\_period2 | Superbowl model for period2 | | #superbowl |
| Sample3\_period3 | Superbowl model for period3 | | #superbowl |
| Sample4\_period1 | Nfl model for period 1 | | #nfl |
| Sample5\_period1 | Nfl model for period 1 | | #nfl |
| Sample6\_period2 | Superbowl model for period 2 | | #superbowl |
| Sample7\_period3 | Nfl model for period 3 | | #nfl |
| Sample8\_period1 | Nfl model for period 1 | | #nfl |
| Sample9\_period2 | Superbowl model for period2 | | #superbowl |
| Sample10\_period3 | Nfl model for period 3 | | #nfl |

Figure: dominant hashtag for the 10 testing files

For all tags the data for 6 hours had been provided. We had to predict the value for next hour. So given the data from hour 1 to hour 6, we had to predict from hour 2 to hour 7.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test File | | Hour 2 | | Hour 3 | | Hour 4 | | Hour 5 | | Hour 6 | | Hour 7 | | Error | |
| Sample1\_period1 | 115.21 | | 50.32 | | 176.80 | | 265.35 | | 461.99 | | 652.02 | | 213.771 | |
| Sample2\_period2 | 614779.9 | | 68409.37 | | 503125 | | 412958 | | 3331221 | | 1806319 | | 1124174.596 | |
| Sample3\_period3 | 510.03 | | 723.36 | | 715.78 | | 628.06 | | 643.21 | | 651.30 | | 197.783 | |
| Sample4\_period1 | 1375.94 | | 562.02 | | 221.95 | | 342.30 | | 134.77 | | 86.02 | | 332.014 | |
| Sample5\_period1 | 491.76 | | 542.83 | | 397.72 | | 308.70 | | 448.62 | | 263.73 | | 253.39 | |
| Sample6\_period2 | 11855.12 | | 108855390 | | 66174686 | | 5643991.7 | | 4233358.1 | | 347051.3 | | 35124214.656 | |
| Sample7\_period3 | 86.61 | | 69.31 | | 60.58 | | 51.63 | | 54.21 | | 68.96 | | 31.343 | |
| Sample8\_period1 | NA | | 57647.17 | | 47250.27 | | 58692.12 | | 72259.96 | | 101448.2 | | 67423.561 | |
| Sample9\_period2 | 907629 | | 936522 | | 790894 | | 750649 | | 1019 | | 895972 | | 715378.320 | |
| Sample10\_period3 | 43.57 | | 41.00 | | 38.55 | | 36.31 | | 35.28 | | 32.25 | | 25.278 | |

Figure : Predicted Value for 7th Hour using Regression Model

Error = Actual – Predicted Vale.  
Note : hour 7 is skipped over here as the data for hour 7 was not available

Part 6 – Fan Base Prediction

In this part, we train a classifier to predict the location of the author, given only the textual context of the tweet. Because often the textual context reveals some information about the author. Recognizing that supporting a sport team has a lot to do with the user location, so, we try to use the textual content of the tweet posted by a user to predict her location.

For this part, we consider all the tweets including #superbowl, by users whose location has been specified as either Washington state or Massachusetts state. [we consider the tweets that include the following substrings in the location field: Seattle, Washington Washington WA Seattle, WA Kirkland, Washington]

We train different classifiers and evaluate their performance. The classifiers used are:

1. Naive Bayes Model
2. Logistic Regression
3. Linear SVM

The following steps are followed:

1. Collect tweets from superbowl
2. Filter out tweets by appropriate location data
3. Create target labels (0: MA 1: WA ) and Balance the datasets
4. Vectorize the tweets:

Since there are lot of common words, the data needs to be preprocessed. For this, first the punctuations are removed, followed by the common stop words. We then find which words share the same stem so that they can be counted together while finding their TF-IDF vectors. To do the latter, we used a SnowBall stemmer (nlkt) to achieve this.

Once the data has been pre-processed, we move on to finding the TF-IDF vector for each term. For this we convert the document into a set of numerical features. This is done using CountVectorizer

1. Truncate twitter data to 50 features using Truncated SVD
2. Perform Feature Scaling for Certain Algorithms Require Nonnegative Values
3. Perform 5-Fold CV to fit different models. The results are given below. **The best accuracy was obtained for Linear SVM as 0.8139.** The performance for Logistic Regression was comparable.

**Multinomial Naive Bayes**

|  |  |
| --- | --- |
| Parameter | Value |
| Average CV- Accuracy | 0.7271 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 (MA) | 0.69 | 0.83 | 0.75 | 3357 |
| 1 (WA) | 0.78 | 0.63 | 0.70 | 3351 |
| Avg/Total | 0.74 | 0.73 | 0.72 | 6708 |

|  |  |
| --- | --- |
| Confusin Matrix | |
| 2771 | 586 |
| 1243 | 2108 |



Figure: Performance evaluation for Multinominal Naïve Bayes

**Logistic Regression:**

|  |  |
| --- | --- |
| Parameter | Value |
| Average CV- Accuracy | 0.8130 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 (MA) | 0.76 | 0.94 | 0.84 | 3357 |
| 1 (WA) | 0.92 | 0.70 | 0.80 | 3351 |
| Avg/Total | 0.84 | 0.82 | 0.82 | 6708 |

|  |  |
| --- | --- |
| Confusin Matrix | |
| 3162 | 195 |
| 1011 | 2340 |



Figure: Performance evaluation for Logistic Regression

**Linear SVM:**

|  |  |
| --- | --- |
| Parameter | Value |
| Average CV- Accuracy | 0.8139 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0 (MA) | 0.76 | 0.95 | 0.84 | 3357 |
| 1 (WA) | 0.93 | 0.69 | 0.80 | 3351 |
| Avg/Total | 0.84 | 0.82 | 0.82 | 6708 |

|  |  |
| --- | --- |
| Confusin Matrix | |
| 3182 | 175 |
| 1023 | 2328 |



Figure: Performance evaluation for Linear SVM