Using Conference Schedules for Dataset Usage Prediction

Ting Li

Advisor: Valentin Kuznetsov

1. Overview

The Compact Muon Solenoid (CMS) is a large general-purpose particle physics detector, built on the Large Hadron Collider (LHC) at CERN in Switzerland and France. The goal of the CMS experiments is to investigate a wide range of physics, including the search for the Higgs boson, extra dimensions, and particles that could make up dark matter. The CMS experiments present challenges not only in terms of the physics to discover and the detector to build and operate, but also in terms of the data volume and the necessary computing resources. Data sets and resource requirements are at least an order of magnitude larger than in previous experiments. The CMS computing system relies on a distributed infrastructure of Grid resources, services and toolkits, to cope with computing requirements for storage, processing and analysis of data provided by the experiments.

It will be beneficial if we can reliably predict the usages of the datasets used in the future experiments, so the data management staff can make enough replicates of the datasets, and deploy them to the storage centers nearby, and thus increase the throughput of data delivery to the researchers.

An approach to the prediction is to binarize the dataset access count in unit time (say, a week) into popularity labels, so that the problem becomes a classification one, and use the attributes from the CMS dataset access data as features. [1] The approach can make sensible prediction in some experiments. The IT engineers, however, are facing ever increasing demand of data delivery, so they continue on searching for ways to achieve better prediction performance.

My project task is to study if and how the CMS conference schedules can be useful for predicting CMS dataset future popularity. This project is

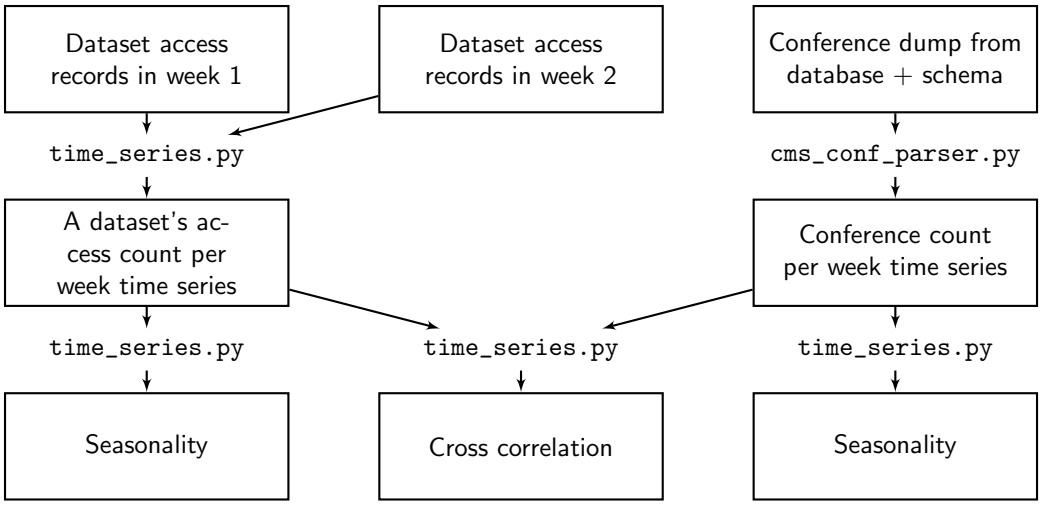

Figure 1: Flowchart of Time Series Generation and Analysis

supervised by Valentin Kuznetsov. I thank him for his guidance and patience.

1.1. Workflow and Design of the Project

My modeling and analysis proceed in two directions: analyze the time series of the weekly conference count and of the weekly access counts to each dataset, and evaluate weekly conference counts as new features added to the classification model.

The diagrams 1 and 2 are the flowcharts of the simplified processes in the two types of modeling and analysis in this project. In the diagrams, program files are shown without being bounded by boxes, and the inputs, intermediates, and outputs of programs are shown inside boxes. The inputs, intermediates, and outputs of a program are simplified, for example, there are much more dataset access files than the two shown for two weeks, and some programs also takes auxiliary inputs besides those shown in the diagrams. The purpose of making the diagrams is to show the design and workflow of the project in a big picture.

The programs which I have written are:

- `cms_conf_parser.py`

  It parses the conference data dump file into a conference count time series.

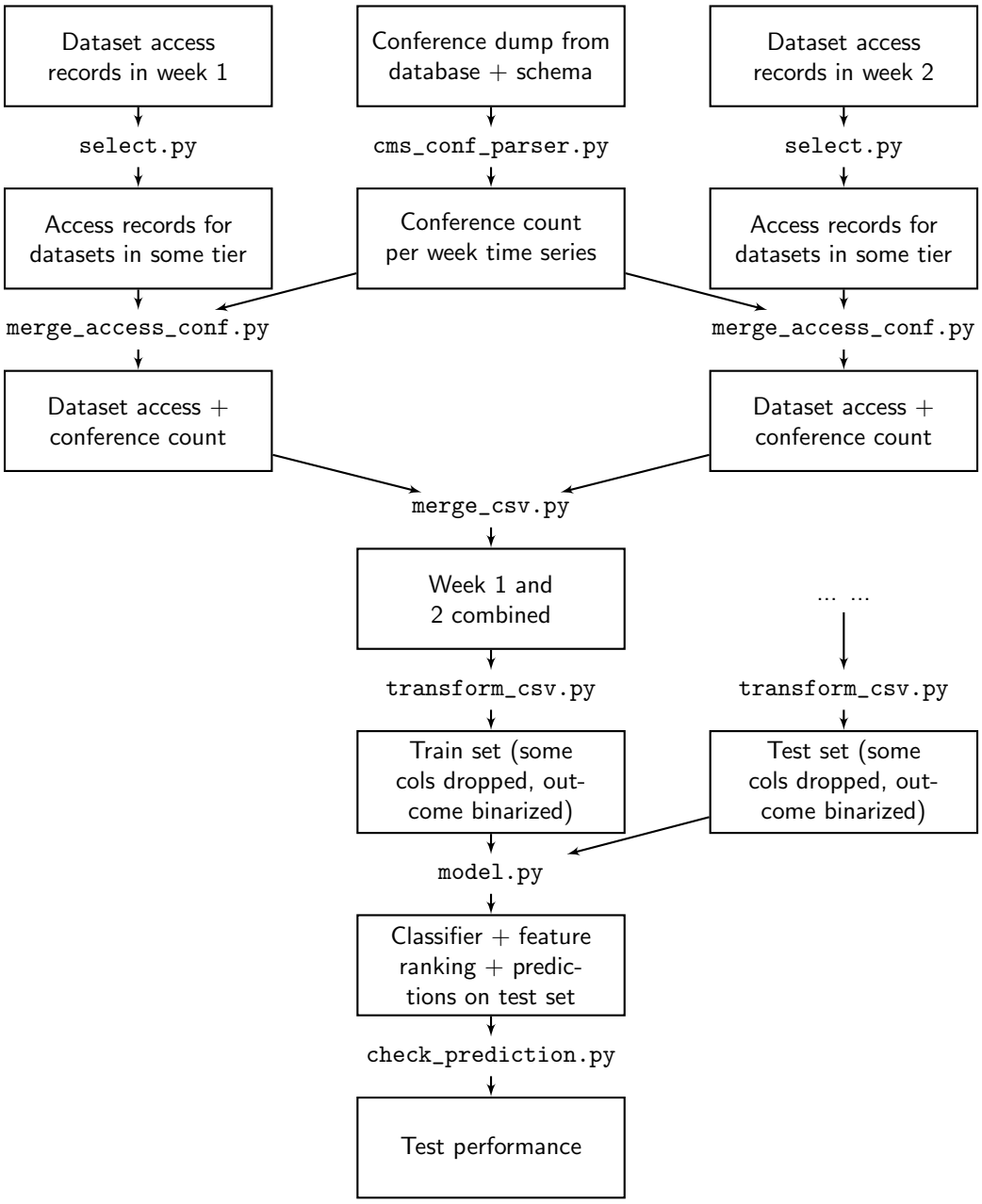

Figure 2: Flowchart of Classification and Train/Test Set Generation

Input:

`--indump`: a csv.gz file for conference data dump from a database

`--inschema`: a plain text file for the schema of the conference data dump file

Output:

`--outdir`: a directory for csv.gz files for conference count per week time series, for conference count for future weeks, and for parsed conference records

- `select.py`

  It selects records from dataset access files whose attribute is some value.

  Input:

  `--indir`: a directory containing the csv.gz files for the input original dataset access data

  `--attr`: an attribute to select by

  `--attrval`: a value of the attribute to select by

  Output:

  `--outdir`: a directory cotaining csv.gz files for the output selected records

- `merge_access_conf.py`

  It add records from conference counts to dataset access records

  Input:

  `--indir`: a directory containing the csv.gz files for the input dataset access data

  `--inconf`: a csv.gz file for the input conference count data

  Output:

  `--outdir`: a directory containing csv.gz files for the output merged data

- `time_series.py`

It generates time series for each dataset from the dataset access data, and analyze the cross correlations and seasonalities for the time series of dataset access and time series of conference count.

Input:

`--indir`: a directory containing csv.gz files for the input dataset access data

`--inconf`: a csv.gz file for the input conference count data

output:

`--outdir`: a directory containing csv.gz files for the output time series of each dataset, and image files for the plots of cross correlation and FFT of the time series

They are available on GitHub [2].

The other programs were written by Valentin [3]. I slighly modified his `model.py` for adding independence tests between each feature and the outcome, changing the split ratio of train and validation sets, fixing the random seeds, and changing the number of feature in the feature importance output.

2. Preparing Datasets

2.1. CMS Conference Data

The CMS conference data are provided to me in a file, dumped from querying a database. It contains a collection of conference records. Each record represents a conference, and consists of the conference's ID, name, cateogry, description, held time, held location, website link, etc.

I perform the following processings on the conference data dump file, to generate the data needed for modeling and analysis:

- Parse the text in the conference data file into a list of dictionaries in Python, according to the conference attribute schema. This is implemented in `cms_conf_parser.py`. The schema provides each attribute's name, type, and length if it is string valued. In the conference data file, each record occupies multiple lines, and records are separated by a blank line. The fields of a record are separated by a comma or a new line, and each field may have comma(s) inside, and may even be blank i.e. missing. The parsing of the dump file according to the schema

is implemented by RegEx pattern matching. Each dicitonary in the list represents a conference record (or schedule), with keys being the conference attributes.

- Group the conference records by their held times in terms of weeks, and count the conferences in each week. This is implemented in `cms_conf_parser.py`. The group by week is implemented by creating a dictionary, with a key being a string of a week, and its value being a list of conference records falling into that week. The output is a file, containing a time series of conference count per week over the weeks from the beginning of 2006 to mid September 2015 (505 weeks in total). Note that a week here is not a calendar year. The first day of a year is always the start of the first week of the year, and the last week of a year may contain more than 7 days.

2.2. CMS Dataset Data

The CMS dataset access data are provided to me as a collection of files.

Each file stores the records of datasets accessed in a week. The file is also named by the start and end dates of the week, for example, `dataframe-20130101-20130107.csv.gz` and there are no timestamp information in the file content.

Each record in the file is for a dataset accessed in the week, and the fields of each record are the information about the dataset (such as id, size) and the usage of the dataset during a week (such as number of access, cpu time). [1] has explanations for the attributes of the record fields.

The contents of the files are organized in csv file format, and hence easier to parse than the files for conference data. Each record occupies a single line, and the records are separated by a newline character. The fields of a record are spearated by a comma.

I perform the following processings on the dataset access files, to generate the data needed for modeling and analysis:

- Extract dataset access records for datasets belonging to Tier 2 and to Tier 3. This is implemented by list comprehension in `select.py`. Note that tiers are levels in the hierachy of CMS distributed computing infrastructure. Tier sites store datasets. There are certain tiers whose sites store datasets dedicated to software testing, and the access to those datasets are unrelated to their usage by physicists. Thus in our project, those datasets are of little interest, and I use only datasets from

Tier 2 (reconstructed data) and Tier 3 (analysis data). The datasets used in this report are from tier 2, and the processing, modeling and analysis for datasets from tier 3 are similar.

- Group the records by datasets, and extract dataset access count per week. This is implemented in `time_series.py`. For each dataset, I group the records of the dataset in all the dataframe files, by using a dictionary, with each key being a dataset and its value being a list of the records of the dataset. A key identifies the dataset to which a record belongs, by the `dataset` and `dbs` attributes of the record. Then I extract the values of `naccess` attribute from the records of the dataset. The `naccess` attribute in a dataset access record is the number of accesses to a dataset, reported by PopularityDB The output is a time series of access count per week over weeks for each dataset.

2.3. Merge Conference Data to Dataset Access Data

In the following section for modeling the problem as a classification one, we merge the conference data to the dataset data. We suspect that the future conference shedules may influence users' current overall access frequency to the datasets, and because they are scheduled for the future, their information is available for predicting future accesses to the datasets.

To each dataset access record, I add the following new fields: the conference counts in the same week as the dataset access record, and in some future weeks. For a record in the dataset access data, the way to find the corresponding record in the conference data is to match their timestamps. Note that the timestamps of their records do not store in the same way however. The timestamp of a record in a dataset file is in the filename, while the timstamp of a record in the conference count file is in a field of the record. Since the timestamps are represented as strings, the matching between the timestamps is a pattern matching problem. This is implemented in `merge_access_conf.py`.

3. Modeling and Analysis for Predicting Dataset Usage

As mentioned in the Overview section, my task is to study whether and how conference data can be used, possibly with CMS dataset data, for predicting future dataset usage. My study has been going in two directions, which differ in whether and how we take into account the time stamps of the CMS dataset records and of conference schedules.

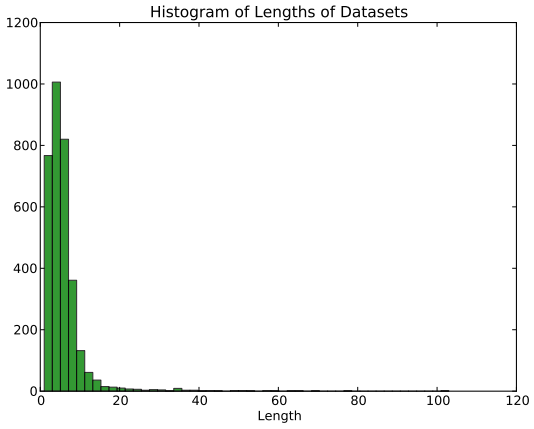

Figure 3: Histogram of lengths of dataset access count series

3.1. Time Series Analysis

In this direction, I analyze the relation between two time series indexed by weeks: conference count per week, and dataset access count per week (given as values of the `naccess` attribute from CMS dataset records).

I consider the time series for dataset access count, one dataset at a time.

There are 3279 datasets in Tier 2, and I do not consider them all for the following reasons.

Different datasets have different numbers of records. For time series analysis purpose, I only consider those datasets which have more than 10 records. There are 237 of them. The maximum length of a dataset is 103, the minimum is 1, and the median is 5, and the standard deviation is 5.59. The histogram of dataset lengths is in Figure 3.

The `naccess` attribute of some datasets are constan (e.g. 0), which makes cross-correlation become invalid, and fourier transform to be a single spike at frequency 0. Prediction of `naccess` of such datasets is also trivial. Therefore I remove such datasets. There are 2973 datasets whose `naccess` attributes are constant, and 2959 of them are constantly 0. The maximum, minimum, median and standard deviation of variances of `naccess` attribute values among the datasets are $826207739.67, 0, 0$, and 22326593.86.

After the selection of datasets by the above two criterions, there are 125 datasets remaining for the time series analysis.

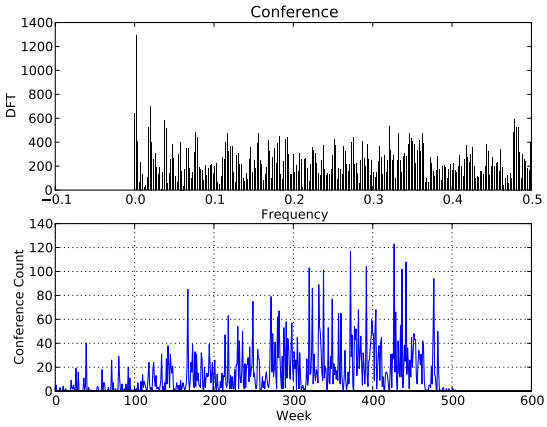

Figure 4: DFT of conference count per week time series

The records of a dataset can be missing for some weeks between its existing records in other weeks. For the purpose of time series analysis, I deal with the missing values by filling with 0 for `naccess` in missing weeks.

3.1.1. Seasonality

It is our original guess that conference schedules and dataset access may expose seasonalities or periodicities, due to holidays and vacations. If it were true, we then would like to use their seasonalities or periodicities to simplify our modeling.

Seasonality is not obvious in either the plot of the conference count series or an arbitrary dataset access series however. An alternative way to study seasonality in a time series is to calculate discrete Fourier Transform on the time series by the Fast Fourier Transform algorithm, and search for significant spikes that represent the frequencies of seasonalities. I calculate the FFT of a time series using numpy.fft.rfft().

Figure 4 is the plot of the DFT of the conference count series, and 5 and 6 the DFT of some dataset access series.

A time series usually has more or less noises, which leads to spurious spikes in its DFT series. Picking out the highest spikes requires smoothing the series with trial and error. Thus I prefer visually checking over automatically choosing the highest spikes.

In the DFT plot of the conference count series, the highest spike occurs at

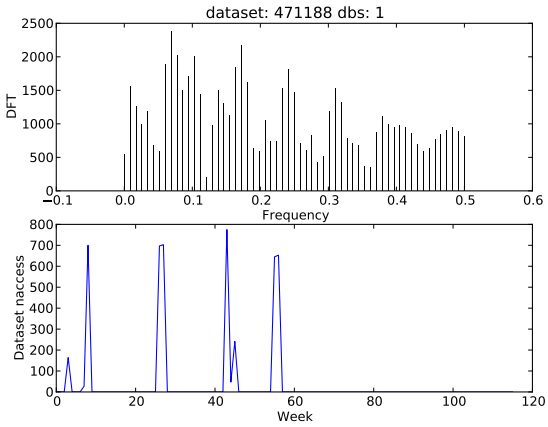

Figure 5: DFT of access count per week time series of a dataset

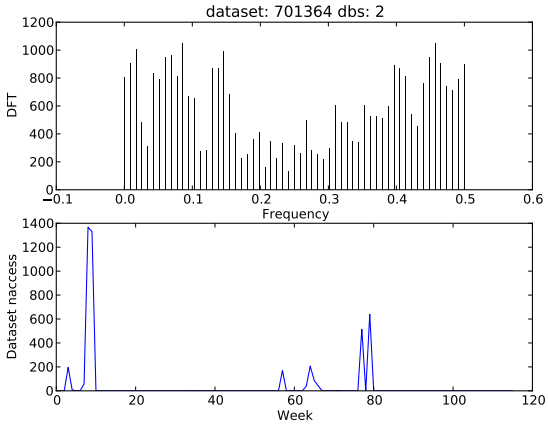

Figure 6: DFT of access count per week time series of a dataset

the first frequency (i.e. frequency 1/505, where 505 is the length of the time series in weeks) , which does not correspond to an actual time period because it corresponds to the whole time period of the time series (505 weeks from the beginning of 2013 to the middle of September 2015). The second highest spike occurs at the 10th fequency (i.e. frequency 10/505), which corresponds to a time period of $505/10 = 50.5$ weeks, roughly a year. This confirms our intial guess of seasonality due to holidays. Since the spike at the first frequency is so dominant, and the second highest spike is not much higher than the other spikes, the time series itself exhibit only slightly periodicity from the second highest spike.

In the DFT plot of the access series of the dataset (471188,1), the highest spike occurs at the 8th frequency (i.e. frequency 8/116, where 116 is the length of the time series in weeks), which corresponds to a time period of $116/8 = 15$ weeks. This coincides with the observation of a periodicity between 10 and 20 weeks from the plot of the time series itself.

In the DFT plot of the access series of the dataset (701364,2), there is no spike significantly higher than the others. So there seems no significant seasonality in the dataset access series.

The DFT plot varies from dataset to dataset, and there seems not guruantee to find seasonality in them.

Neither is it clear yet how to relate the seasonalities of the dataset access series with that of the conference count series. If both the conference count series and an individual dataset access count series are both periodic, we may limit our study to a time interval whose length is the common least multiple of their periods, instead of the entire time intervals where the two time serires were created/defined.

3.1.2. Cross Correlation

The cross correlation between the time series of a data set's `naccess` attribute, and the conference count time series, over lags ranging between -50 and 50 with step 1. At each lag, the cross correlation is calculated by scipy.stats.stats.pearsonr(). Besides cross correlation, it also returns a p-value for testing a null hypothesis that two series are uncorrelated. The plots 7, 8 and 9 are examples of the `naccess` series of some dataset, the conference count series, and the cross correlation between them.

The plots show that the lag where the highest cross corrleation in magnitude occurs varies with different datasets, and can be either positive lags (e.g. the plot for datasets `(471188,1)` and `(701364,2)`), or negative lags

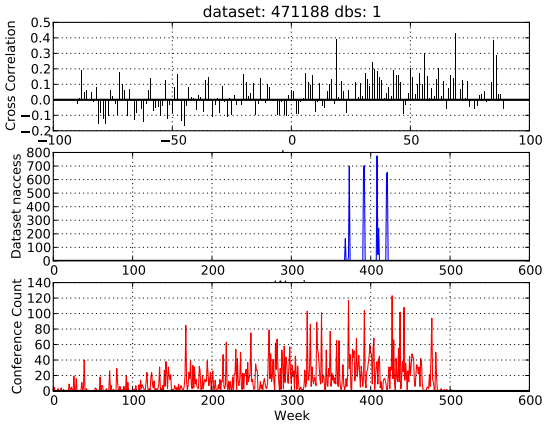

Figure 7: Cross correlation between the conference count series and the access count series of a dataset

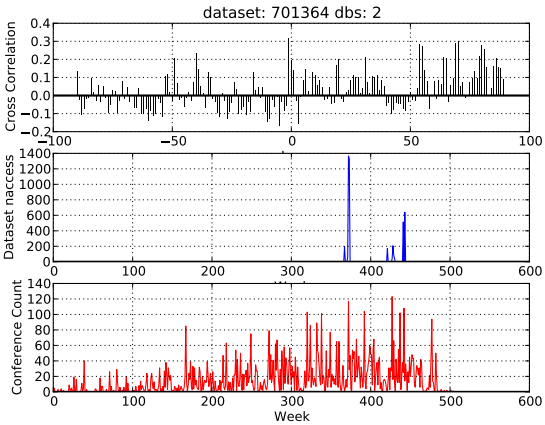

Figure 8: Cross correlation between the conference count series and the access count series of a dataset

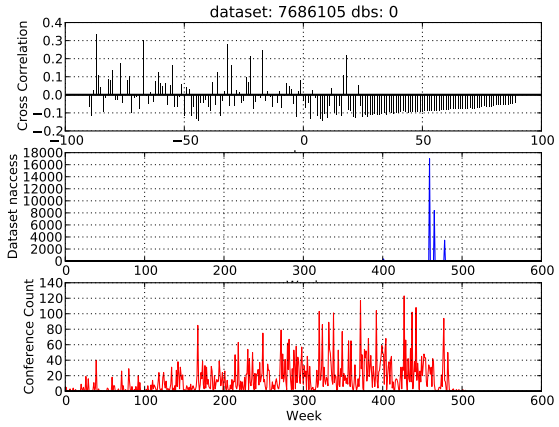

Figure 9: Cross correlation between the conference count series and the access count series of a dataset

(e.g. the plot for dataset `(7686105,0)`). Cross correlation peaking at a positive lag means that future conference schedules can lead the current dataset access, while peaking at a negative lag means that past conferences can still have residual influence on the current dataset access.

The plot 10 is the histogram of such lags with p-values smaller than 0.05 (indicating the uncorrelated null is rejected with significance level 0.05), which shows that cross correlation achieves its maximum most probably around 75 lags in weeks (roughly one year and a half), and at the lags $65, 85, 40, 50, 60$, and 5 with decreasing frequencies:

Based on the cross correlation analysis, for a given dataset, we can build a prediction model by regressing the dataset access count in each week on future conference counts in some weeks away by the lags chosen from the cross correlation analysis). The potential draw back of such modeling may be due to:

- We assume the relation between dataset access count and future conference counts are the same over the time, which is something like or similar to stationarity: the distribution of a week's dataset access count given future weeks' conference counts is the same across the weeks. After building such models, we lose track of time. Without the stationary assumption, it is unclear yet how to build a model that forecasts the

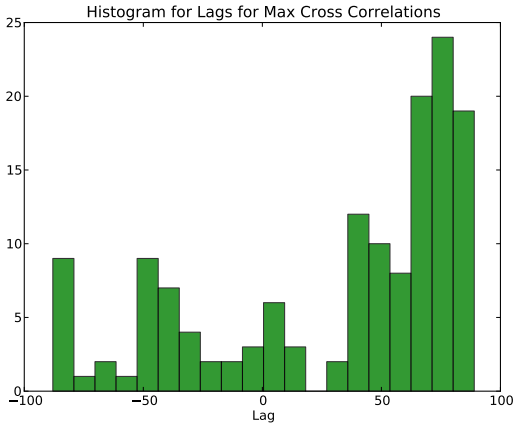

Figure 10: Histogram of lags with maximum cross correlation

dataset access count series from the conference count series, and captures the nonstationary relation between the two time series. It will be good to test the stationary assumption nonetheless.

- It does not make use of the other attributes from CMS dataset data, except using the dataset access count as the outcome, and the dataset access timestamp for matching with the future conference counts.

3.2. Classification

In this subsection, we simplify the quantitative outcome, i.e. the weekly dataset access count, to a qualitative popularity class label, by binarizing the counts at a threshod 100. This changes the problem from a regression one to a classification one, potentially increasing the generalization ability of the trained model.

Moreover, we make the following assumptions:

- Seasonality: Assume the conference count series and each dataset popularity label series are periodic, say, with 1 year (52 weeks) being their common (least) multiple. We choose to train on the data from the May 2013 to April 2014, and test on the data in the following week after April 2014.

- Stationarity: the relation between the dataset popularity label, and conference count and other dataset attributes from CMS dataset data is stationary, i.e. does not change over time. So we drop the timestamps from the data after combining the conference counts and the dataset access records.

Some notes:

- When I initially chose the feautres used in the classification problem, I would like to consider all the available information, and might later perform feature selection as needed. Thus the features are the future conference counts weeks away by lags chosen from the cross correlation analysis of time series, and the attributes in CMS dataset data that are available at the time of prediction.

  The conference count per week can change dramatically, so instead of using them directly, I use more stable features which are the accumulated future conference counts up to $1, 2, 4, 6, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65,$ and 70 weeks. The number of the weeks are chosen, somehow based on the lags at which the cross correlation between conference count and dataset access count achieve high (see the previous section).

  Variables whose values are not available before the time to predict for, should be dropped from features. These have been singled out by Valentin in his demo examples of using the his programs. For example, `naccess` must obviously be dropped. So are `nusers` (the number of users*days to a dataset, reported by PopularityDB), and `totcpu` (the number of cpu hours to accessed dataset, reported by PopularityDB).

- Classifier, feature transformation, feature selection and ranking may depend on feature type (categorical, ordered, numerical, ...), and may not work with some feature types. Some features from CMS dataset data seem to be categorical, e.g. `dataset`, `dbs`, `rel*`, and more (todo). In the data files, all the features (including categorical ones) are coded in numerical values, which can be problematic if not handled correctly.

- We are intersted more in popular class than the unpopular one, so we choose classification performance measures that more focus on the positive class: accuracy, precision, recall and $F1$ scorers.

There are different ways to rank the features in a classification problem.

3.2.1. Importance Measurements Returned by Random Forest Classifier

When a random forest classifier make predictions on a test set, the relative rank (i.e. depth) of a feature used as a decision node in a tree can be used to assess the relative importance of that feature with respect to the predictability of the target variable. Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples. The expected fraction of the samples they contribute to can thus be used as an estimate of the relative importance of the features. [4]

I train random forest classifiers by calling the CMS program `model.py`, which in turn calls the learning algorithm implemented in the Scikit-Learn library by the class `sklearn.ensemble.RandomForestClassifier`.

Note:

- I use the default complexity settings of a random forest classifier based on the empirical recommendations in the library's user guide. In the default settings, a random forest has 10 random trees, each tree is allowed to be contructed without constraints on its complexity (depth), and each split at a node is determined by a randomly chose subset of features, the number of which is the square root of the number of all the features.

  Thanks to the randmization in constructing each tree, and averaging the probabilistic predictions of the individual trees, the tendency of a trained random forest classifier to overfit is generally small.

  Thus I skip the validation step to tune the complexity of the forest. Then the setting of validation set is not necessary, and it also reduce the actual training set. I skip validation by setting the validate size `split=0` in `model.py`.

- A random forest classifier trained from a train set has already arranged the features at each node based on the train set, so the calculation of feature importance by the trained random forest classifier should be on a different data set (here a test set).

- To make the experiments reproducible, I fix the random seeds in the program `model.py`, which appear in the `random_state` parameter to both the constructor function `sklearn.ensemble.RandomForestClassifier()`, and to the splitting datset function `train_test_split()`. Since I skip

validation by setting the validate size `split=0`, the `random_state` parameter to `train_test_split()` does not matter.

- Decision trees and therefore random forest can take categorical features. Scikit-learn does not seem to directly support for categorical features in trees or forests [5]. If categorical features are coded as id numbers, this imposes a non-existing ordering on the categorical values, and the random forest training algorithm will rely on that ordering, and the trained random forest classifier will be misleading in terms of interpretation and more overfit to the training data.

- It may also be misleading to apply numerical feature transformation (such as standardization of features to have zero mean and unit variance) on categorical features that are numerically coded. The random forest training algorithm theoretically does not depend on standardization, because standardization does not change the ordering between the feature values. Feature transformations can change the features undesirably for numerically coded categorical features, It seems better not to do standardization in preprocessing. (I tried to set `scaler=None` (or just not specify the option) as a command line argument to the program `model`, but the program will not predict on the test set.)

3.2.2. Feature Ranking by Independence Testing

Alternatively, we can rank the features, by testing the independence between each feature and the popularity outcome, and then ordering their p-values (the smaller the p value for a feature, the more confidence we will have to reject the null hypothesis of independence between the feature and the outcome).

I use `sklearn.feature_selection.chi2()` and `sklearn.feature_selection.f_classif()` in the Scikit-Learn library to perform the test. They implement two independence tests: the Chi-squared independence test and the ANOVA F test respectively.

Some notes:

- Since the independence tests are not part of the random forest learning algorithm, we can perform the tests on the training set.

- ANOVA F test can only work between a numerical feature and a categorical outcome, because it calculates sample mean and variance of a

feature. So the p values returned by the test for categorical features that are numerically coded may be unreliable.

3.2.3. Experiments and Analysis

In the output of the commands (see Appendix A), the two independence tests show that

- Future conference counts accumulated no less than 40 weeks ahead mostly have p-values less than 0.05, while those within 40 weeks ahead mostly have bigger than 0.05 p-values. This means that future conference counts likely rank higher if they are accumulated into futher future. However it is hard to use conference data that are more into the future (greate than 70 weeks), because the CMS dataset data are provided only between Jan 2013 and May 2015, the CMS conference data are provided as late as Sept 2015, the records of these two data are matched based on their timestamps, and the experiment has already used the data from May 2013 to April 2014.
- Most of the attributes from the CMS dataset data rank higher than most of the future conference count features. This implies that the future conference counts are not as important to the classification problem as many attributes from CMS dataset data.

The feature importances from the trained random forest classifier also show the similar things, and some further observations are:

- Some of `rel2_N`, `rel3_N` features are the most important. These features represents releases of datasets, and therefore their values are ordered. It is reasonable that datasets with longer release histories are more likely on high demand, and therefore will be more popular in the future as well.
- `cpu` and `proc_evts` are also important. If `cpu` and `proc_evts` are measured only after datasets are accessed, then they may not be available at the time of predicting dataset future popularity, and should be dropped as well. (They were not dropped in the demo example.)

The performance of the trained random forest classifier on the test set is almost perfect.

```
Score metric (accuracy_score): 0.993377483444
Score metric (precision_score): 1.0
Score metric (recall_score): 0.909090909091
Score metric (f1_score): 0.952380952381
```

For comparison, when the features are only those from the CMS dataset access data without the future conferenct counts, the performace is perfect (see Appendix B):

```
Score metric (accuracy_score): 1.0
Score metric (precision_score): 1.0
Score metric (recall_score): 1.0
Score metric (f1_score): 1.0
```

When the features are only the future conference counts (plus `dataset` and `dbs`, which must be used by `model.py`), the performace falls backsomehow (see Appendix C):

```
Score metric (accuracy_score): 0.960264900662
Score metric (precision_score): 0.777777777778
Score metric (recall_score): 0.636363636364
Score metric (f1_score): 0.7
```

The trained random forest classifier ranks `dataset` and `dbs` higher than the future conference counts This again implies that the future conference counts are not as important as the attributes from the dataset access data.

4. Conclusion

In this project, we study the influence of the future conference counts on prediciting the dataset popularity.

The experiments show that the future conference counts can provide insights into understanding the problem and data, but are not as important to the classification as many attributes from CMS dataset data.

Alternative ways to use conference counts in the classification problem, and modelling methods other than classification (such as regression or modelling between time series) may be worth exploration.

Appendix A. Output when using both future conference counts and dataset access attributes as features

```
[tili@lxplus0079 testset]$ model --learner=RandomForestClassifier --idcol=id --t
RandomForestClassifier(bootstrap=True, compute_importances=None,
criterion='gini', max_depth=None, max_features='auto',
max_leaf_nodes=None, min_density=None, min_samples_leaf=1,
min_samples_split=2, n_estimators=10, n_jobs=1,
oob_score=False, random_state=123, verbose=0)
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/data/classification/tier2/mer
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/DCAFPIlot/slc6_amd64_gcc481/e
fprob is deprecated in scipy 0.14, use stats.f.sf or special.fdtrc instead

warnings.warn(depdoc, DeprecationWarning)
feature ranking by ANOVA F test:
1. feature selection test p-value 0.000000, feature cpu
2. feature selection test p-value 0.000000, feature rel2_9
3. feature selection test p-value 0.000000, feature rel3_0
4. feature selection test p-value 0.000000, feature rel3_1
5. feature selection test p-value 0.000000, feature rel3_10
6. feature selection test p-value 0.000000, feature rel3_11
7. feature selection test p-value 0.000000, feature rel3_12
8. feature selection test p-value 0.000000, feature rel3_13
9. feature selection test p-value 0.000000, feature rel3_14
10. feature selection test p-value 0.000000, feature rel3_16
11. feature selection test p-value 0.000000, feature rel3_17
12. feature selection test p-value 0.000000, feature rel3_18
13. feature selection test p-value 0.000000, feature rel3_19
14. feature selection test p-value 0.000000, feature rel3_2
15. feature selection test p-value 0.000000, feature rel2_8
16. feature selection test p-value 0.000000, feature rel3_20
17. feature selection test p-value 0.000000, feature rel3_23
18. feature selection test p-value 0.000000, feature rel3_24
19. feature selection test p-value 0.000000, feature rel3_3
20. feature selection test p-value 0.000000, feature rel3_4
21. feature selection test p-value 0.000000, feature rel3_5
22. feature selection test p-value 0.000000, feature rel3_6
23. feature selection test p-value 0.000000, feature rel3_7
```

24. feature selection test p-value 0.000000, feature rel3_8
25. feature selection test p-value 0.000000, feature rel3_9
26. feature selection test p-value 0.000000, feature relt_0
27. feature selection test p-value 0.000000, feature relt_1
28. feature selection test p-value 0.000000, feature relt_2
29. feature selection test p-value 0.000000, feature size
30. feature selection test p-value 0.000000, feature rel3_22
31. feature selection test p-value 0.000000, feature rel2_7
32. feature selection test p-value 0.000000, feature rel3_15
33. feature selection test p-value 0.000000, feature rel2_5
34. feature selection test p-value 0.000000, feature creator
35. feature selection test p-value 0.000000, feature dataset
36. feature selection test p-value 0.000000, feature dbs
37. feature selection test p-value 0.000000, feature nevt
38. feature selection test p-value 0.000000, feature rel2_6
39. feature selection test p-value 0.000000, feature nlumis
40. feature selection test p-value 0.000000, feature nrel
41. feature selection test p-value 0.000000, feature proc_evts
42. feature selection test p-value 0.000000, feature rel1_0
43. feature selection test p-value 0.000000, feature rel1_1
44. feature selection test p-value 0.000000, feature rel1_2
45. feature selection test p-value 0.000000, feature rel1_3
46. feature selection test p-value 0.000000, feature nfiles
47. feature selection test p-value 0.000000, feature rel1_5
48. feature selection test p-value 0.000000, feature rel1_4
49. feature selection test p-value 0.000000, feature rel2_3
50. feature selection test p-value 0.000000, feature rel2_2
51. feature selection test p-value 0.000000, feature rel2_11
52. feature selection test p-value 0.000000, feature rel2_10
53. feature selection test p-value 0.000000, feature rel2_1
54. feature selection test p-value 0.000000, feature rel2_0
55. feature selection test p-value 0.000000, feature rel2_4
56. feature selection test p-value 0.000000, feature rel1_7
57. feature selection test p-value 0.000000, feature rel1_6
58. feature selection test p-value 0.000000, feature nblk
59. feature selection test p-value 0.000000, feature rel3_26
60. feature selection test p-value 0.000000, feature era
61. feature selection test p-value 0.000000, feature rel3_25

```
62. feature selection test p-value 0.000000, feature primds
63. feature selection test p-value 0.000000, feature 60wk
64. feature selection test p-value 0.000000, feature 65wk
65. feature selection test p-value 0.000000, feature 70wk
66. feature selection test p-value 0.000000, feature dtype
67. feature selection test p-value 0.000000, feature 55wk
68. feature selection test p-value 0.000001, feature 50wk
69. feature selection test p-value 0.000167, feature rel3_21
70. feature selection test p-value 0.001715, feature 30wk
71. feature selection test p-value 0.004668, feature 35wk
72. feature selection test p-value 0.033383, feature 45wk
73. feature selection test p-value 0.084018, feature parent
74. feature selection test p-value 0.309144, feature 10wk
75. feature selection test p-value 0.431895, feature 0wk
76. feature selection test p-value 0.480560, feature 1wk
77. feature selection test p-value 0.508494, feature 25wk
78. feature selection test p-value 0.580045, feature 4wk
79. feature selection test p-value 0.849305, feature 20wk
80. feature selection test p-value 0.859428, feature 15wk
81. feature selection test p-value 0.929217, feature procds
82. feature selection test p-value 0.945397, feature 2wk
83. feature selection test p-value 1.000000, feature 6wk
84. feature selection test p-value 1.000000, feature 40wk
85. feature selection test p-value -nan, feature tier
feature ranking by Chi Squared test:
1. feature selection test p-value 0.000000, feature cpu
2. feature selection test p-value 0.000000, feature rel2_9
3. feature selection test p-value 0.000000, feature rel3_0
4. feature selection test p-value 0.000000, feature rel3_1
5. feature selection test p-value 0.000000, feature rel3_10
6. feature selection test p-value 0.000000, feature rel3_11
7. feature selection test p-value 0.000000, feature rel3_12
8. feature selection test p-value 0.000000, feature rel3_13
9. feature selection test p-value 0.000000, feature rel3_14
10. feature selection test p-value 0.000000, feature rel3_16
11. feature selection test p-value 0.000000, feature rel3_17
12. feature selection test p-value 0.000000, feature rel3_18
13. feature selection test p-value 0.000000, feature rel3_19
```

14. feature selection test p-value 0.000000, feature rel3_2
15. feature selection test p-value 0.000000, feature rel2_8
16. feature selection test p-value 0.000000, feature rel3_20
17. feature selection test p-value 0.000000, feature rel3_23
18. feature selection test p-value 0.000000, feature rel3_24
19. feature selection test p-value 0.000000, feature rel3_3
20. feature selection test p-value 0.000000, feature rel3_4
21. feature selection test p-value 0.000000, feature rel3_5
22. feature selection test p-value 0.000000, feature rel3_6
23. feature selection test p-value 0.000000, feature rel3_7
24. feature selection test p-value 0.000000, feature rel3_8
25. feature selection test p-value 0.000000, feature rel3_9
26. feature selection test p-value 0.000000, feature relt_0
27. feature selection test p-value 0.000000, feature relt_1
28. feature selection test p-value 0.000000, feature relt_2
29. feature selection test p-value 0.000000, feature size
30. feature selection test p-value 0.000000, feature rel3_22
31. feature selection test p-value 0.000000, feature rel2_7
32. feature selection test p-value 0.000000, feature rel3_15
33. feature selection test p-value 0.000000, feature rel2_5
34. feature selection test p-value 0.000000, feature creator
35. feature selection test p-value 0.000000, feature dataset
36. feature selection test p-value 0.000000, feature dbs
37. feature selection test p-value 0.000000, feature era
38. feature selection test p-value 0.000000, feature nblk
39. feature selection test p-value 0.000000, feature nevt
40. feature selection test p-value 0.000000, feature rel2_6
41. feature selection test p-value 0.000000, feature nlumis
42. feature selection test p-value 0.000000, feature parent
43. feature selection test p-value 0.000000, feature primds
44. feature selection test p-value 0.000000, feature proc_evts
45. feature selection test p-value 0.000000, feature rel1_0
46. feature selection test p-value 0.000000, feature rel1_1
47. feature selection test p-value 0.000000, feature nfiles
48. feature selection test p-value 0.000000, feature rel1_3
49. feature selection test p-value 0.000000, feature rel1_2
50. feature selection test p-value 0.000000, feature rel2_3
51. feature selection test p-value 0.000000, feature rel2_2

```
52. feature selection test p-value 0.000000, feature rel2_11
53. feature selection test p-value 0.000000, feature rel2_10
54. feature selection test p-value 0.000000, feature rel2_1
55. feature selection test p-value 0.000000, feature rel2_0
56. feature selection test p-value 0.000000, feature rel1_7
57. feature selection test p-value 0.000000, feature rel2_4
58. feature selection test p-value 0.000000, feature rel1_6
59. feature selection test p-value 0.000000, feature rel1_4
60. feature selection test p-value 0.000000, feature rel1_5
61. feature selection test p-value 0.000000, feature procds
62. feature selection test p-value 0.000000, feature 70wk
63. feature selection test p-value 0.000000, feature 65wk
64. feature selection test p-value 0.000000, feature 60wk
65. feature selection test p-value 0.000000, feature nrel
66. feature selection test p-value 0.000000, feature 55wk
67. feature selection test p-value 0.000000, feature 30wk
68. feature selection test p-value 0.000000, feature rel3_26
69. feature selection test p-value 0.000000, feature rel3_25
70. feature selection test p-value 0.000000, feature 50wk
71. feature selection test p-value 0.000000, feature 35wk
72. feature selection test p-value 0.000000, feature 10wk
73. feature selection test p-value 0.000120, feature 45wk
74. feature selection test p-value 0.000184, feature rel3_21
75. feature selection test p-value 0.000192, feature 0wk
76. feature selection test p-value 0.000801, feature 1wk
77. feature selection test p-value 0.004897, feature 4wk
78. feature selection test p-value 0.008272, feature 25wk
79. feature selection test p-value 0.381273, feature 20wk
80. feature selection test p-value 0.385936, feature 15wk
81. feature selection test p-value 0.408017, feature 40wk
82. feature selection test p-value 0.412632, feature dtype
83. feature selection test p-value 0.736402, feature 2wk
84. feature selection test p-value 0.966743, feature 6wk
85. feature selection test p-value 1.000000, feature tier
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/data/classification/tier2/mer
Feature ranking:
1. importance 0.093759, feature rel3_9
2. importance 0.089306, feature rel3_5
```

3. importance 0.086117, feature relt_2
4. importance 0.084869, feature rel2_3
5. importance 0.080612, feature rel3_0
6. importance 0.080032, feature relt_0
7. importance 0.079352, feature rel2_5
8. importance 0.076952, feature rel3_2
9. importance 0.075827, feature rel1_3
10. importance 0.071182, feature rel3_17
11. importance 0.024409, feature proc_evts
12. importance 0.021641, feature cpu
13. importance 0.011592, feature rel1_5
14. importance 0.008654, feature nfiles
15. importance 0.007422, feature size
16. importance 0.006638, feature rel1_6
17. importance 0.004823, feature nblk
18. importance 0.004628, feature rel3_7
19. importance 0.004028, feature rel3_8
20. importance 0.003601, feature rel3_1
21. importance 0.003508, feature rel2_1
22. importance 0.002963, feature nlumis
23. importance 0.002915, feature rel2_0
24. importance 0.002912, feature rel3_3
25. importance 0.002881, feature 25wk
26. importance 0.002853, feature 10wk
27. importance 0.002794, feature rel1_4
28. importance 0.002709, feature rel2_2
29. importance 0.002637, feature rel1_2
30. importance 0.002509, feature 70wk
31. importance 0.002487, feature 65wk
32. importance 0.002467, feature rel3_13
33. importance 0.002433, feature rel3_4
34. importance 0.002317, feature 0wk
35. importance 0.002266, feature 55wk
36. importance 0.002211, feature 50wk
37. importance 0.002163, feature nevt
38. importance 0.002159, feature 35wk
39. importance 0.002114, feature rel3_6
40. importance 0.002062, feature rel3_11

41. importance 0.002027, feature parent
42. importance 0.001964, feature rel2_4
43. importance 0.001724, feature dataset
44. importance 0.001684, feature 1wk
45. importance 0.001623, feature rel1_0
46. importance 0.001620, feature 40wk
47. importance 0.001566, feature era
48. importance 0.001518, feature relt_1
49. importance 0.001487, feature creator
50. importance 0.001290, feature 30wk
51. importance 0.001289, feature rel1_7
52. importance 0.001251, feature 4wk
53. importance 0.001217, feature procds
54. importance 0.001200, feature primds
55. importance 0.001112, feature 2wk
56. importance 0.001004, feature rel2_9
57. importance 0.000995, feature 60wk
58. importance 0.000896, feature rel3_10
59. importance 0.000884, feature 20wk
60. importance 0.000830, feature 45wk
61. importance 0.000772, feature dbs
62. importance 0.000760, feature rel3_12
63. importance 0.000758, feature nrel
64. importance 0.000735, feature rel2_6
65. importance 0.000726, feature rel3_15
66. importance 0.000638, feature 15wk
67. importance 0.000575, feature rel3_14
68. importance 0.000540, feature 6wk
69. importance 0.000491, feature rel3_16
70. importance 0.000021, feature rel2_8
71. importance 0.000000, feature rel2_11
72. importance 0.000000, feature rel3_25
73. importance 0.000000, feature rel2_7
74. importance 0.000000, feature tier
75. importance 0.000000, feature rel1_1
76. importance 0.000000, feature rel2_10
77. importance 0.000000, feature rel3_18
78. importance 0.000000, feature rel3_19

```
  79. importance 0.000000, feature rel3_20
  80. importance 0.000000, feature rel3_21
  81. importance 0.000000, feature dtype
  82. importance 0.000000, feature rel3_22
```

Appendix B. Output when using only dataset access attributes as features

```
[tili@lxplus0079 testset]$ model --learner=RandomForestClassifier --idcol=id --tar
> -newdata="$datasetdir"/testset/transform/dataframe-20140507-20140513.csv.gz --pr
> ecall,f1

RandomForestClassifier(bootstrap=True, compute_importances=None,
            criterion='gini', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_density=None, min_samples_leaf=1,
            min_samples_split=2, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=123, verbose=0)
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/data/classification/tier2/tier2
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/DCAFPIlot/slc6_amd64_gcc481/ext
fprob is deprecated in scipy 0.14, use stats.f.sf or special.fdtrc instead

  warnings.warn(depdoc, DeprecationWarning)

Feature ranking by ANOVA F test:
1. feature selection test p-value 0.000000, feature cpu
2. feature selection test p-value 0.000000, feature rel2_9
3. feature selection test p-value 0.000000, feature rel3_0
4. feature selection test p-value 0.000000, feature rel3_1
5. feature selection test p-value 0.000000, feature rel3_10
6. feature selection test p-value 0.000000, feature rel3_11
7. feature selection test p-value 0.000000, feature rel3_12
8. feature selection test p-value 0.000000, feature rel3_13
9. feature selection test p-value 0.000000, feature rel3_14
10. feature selection test p-value 0.000000, feature rel3_15
11. feature selection test p-value 0.000000, feature rel3_16
12. feature selection test p-value 0.000000, feature rel3_17
13. feature selection test p-value 0.000000, feature rel3_18
14. feature selection test p-value 0.000000, feature rel3_19
```

15. feature selection test p-value 0.000000, feature size
16. feature selection test p-value 0.000000, feature rel3_2
17. feature selection test p-value 0.000000, feature rel3_22
18. feature selection test p-value 0.000000, feature rel3_23
19. feature selection test p-value 0.000000, feature rel3_24
20. feature selection test p-value 0.000000, feature rel3_3
21. feature selection test p-value 0.000000, feature rel3_4
22. feature selection test p-value 0.000000, feature rel3_5
23. feature selection test p-value 0.000000, feature rel3_6
24. feature selection test p-value 0.000000, feature rel3_7
25. feature selection test p-value 0.000000, feature rel3_8
26. feature selection test p-value 0.000000, feature rel3_9
27. feature selection test p-value 0.000000, feature relt_0
28. feature selection test p-value 0.000000, feature relt_1
29. feature selection test p-value 0.000000, feature relt_2
30. feature selection test p-value 0.000000, feature rel3_20
31. feature selection test p-value 0.000000, feature rel2_7
32. feature selection test p-value 0.000000, feature rel2_8
33. feature selection test p-value 0.000000, feature rel2_5
34. feature selection test p-value 0.000000, feature nevt
35. feature selection test p-value 0.000000, feature nfiles
36. feature selection test p-value 0.000000, feature nlumis
37. feature selection test p-value 0.000000, feature nrel
38. feature selection test p-value 0.000000, feature dataset
39. feature selection test p-value 0.000000, feature proc_evts
40. feature selection test p-value 0.000000, feature rel2_6
41. feature selection test p-value 0.000000, feature creator
42. feature selection test p-value 0.000000, feature rel1_1
43. feature selection test p-value 0.000000, feature rel1_2
44. feature selection test p-value 0.000000, feature rel1_3
45. feature selection test p-value 0.000000, feature rel1_0
46. feature selection test p-value 0.000000, feature rel1_5
47. feature selection test p-value 0.000000, feature rel2_4
48. feature selection test p-value 0.000000, feature rel2_3
49. feature selection test p-value 0.000000, feature rel2_2
50. feature selection test p-value 0.000000, feature rel2_11
51. feature selection test p-value 0.000000, feature rel1_4
52. feature selection test p-value 0.000000, feature rel2_10

```
53. feature selection test p-value 0.000000, feature dbs
54. feature selection test p-value 0.000000, feature rel2_1
55. feature selection test p-value 0.000000, feature rel2_0
56. feature selection test p-value 0.000000, feature rel1_7
57. feature selection test p-value 0.000000, feature rel1_6
58. feature selection test p-value 0.000000, feature nblk
59. feature selection test p-value 0.000000, feature rel3_26
60. feature selection test p-value 0.000000, feature era
61. feature selection test p-value 0.000000, feature rel3_25
62. feature selection test p-value 0.000000, feature primds
63. feature selection test p-value 0.000000, feature dtype
64. feature selection test p-value 0.000167, feature rel3_21
65. feature selection test p-value 0.084018, feature parent
66. feature selection test p-value 0.929217, feature procds
67. feature selection test p-value -nan, feature tier

Feature ranking by Chi Squared test:
1. feature selection test p-value 0.000000, feature cpu
2. feature selection test p-value 0.000000, feature rel2_9
3. feature selection test p-value 0.000000, feature rel3_0
4. feature selection test p-value 0.000000, feature rel3_1
5. feature selection test p-value 0.000000, feature rel3_10
6. feature selection test p-value 0.000000, feature rel3_11
7. feature selection test p-value 0.000000, feature rel3_12
8. feature selection test p-value 0.000000, feature rel3_13
9. feature selection test p-value 0.000000, feature rel3_14
10. feature selection test p-value 0.000000, feature rel3_15
11. feature selection test p-value 0.000000, feature rel3_16
12. feature selection test p-value 0.000000, feature rel3_17
13. feature selection test p-value 0.000000, feature rel3_18
14. feature selection test p-value 0.000000, feature rel3_19
15. feature selection test p-value 0.000000, feature size
16. feature selection test p-value 0.000000, feature rel3_2
17. feature selection test p-value 0.000000, feature rel3_22
18. feature selection test p-value 0.000000, feature rel3_23
19. feature selection test p-value 0.000000, feature rel3_24
20. feature selection test p-value 0.000000, feature rel3_3
21. feature selection test p-value 0.000000, feature rel3_4
```

22. feature selection test p-value 0.000000, feature rel3_5
23. feature selection test p-value 0.000000, feature rel3_6
24. feature selection test p-value 0.000000, feature rel3_7
25. feature selection test p-value 0.000000, feature rel3_8
26. feature selection test p-value 0.000000, feature rel3_9
27. feature selection test p-value 0.000000, feature relt_0
28. feature selection test p-value 0.000000, feature relt_1
29. feature selection test p-value 0.000000, feature relt_2
30. feature selection test p-value 0.000000, feature rel3_20
31. feature selection test p-value 0.000000, feature rel2_7
32. feature selection test p-value 0.000000, feature rel2_8
33. feature selection test p-value 0.000000, feature rel2_5
34. feature selection test p-value 0.000000, feature creator
35. feature selection test p-value 0.000000, feature era
36. feature selection test p-value 0.000000, feature nblk
37. feature selection test p-value 0.000000, feature nevt
38. feature selection test p-value 0.000000, feature nfiles
39. feature selection test p-value 0.000000, feature nlumis
40. feature selection test p-value 0.000000, feature parent
41. feature selection test p-value 0.000000, feature primds
42. feature selection test p-value 0.000000, feature proc_evts
43. feature selection test p-value 0.000000, feature rel2_6
44. feature selection test p-value 0.000000, feature dbs
45. feature selection test p-value 0.000000, feature rel1_1
46. feature selection test p-value 0.000000, feature rel1_0
47. feature selection test p-value 0.000000, feature rel1_3
48. feature selection test p-value 0.000000, feature rel2_4
49. feature selection test p-value 0.000000, feature rel2_3
50. feature selection test p-value 0.000000, feature rel2_2
51. feature selection test p-value 0.000000, feature rel2_11
52. feature selection test p-value 0.000000, feature rel2_10
53. feature selection test p-value 0.000000, feature rel2_1
54. feature selection test p-value 0.000000, feature rel1_2
55. feature selection test p-value 0.000000, feature dataset
56. feature selection test p-value 0.000000, feature rel2_0
57. feature selection test p-value 0.000000, feature rel1_7
58. feature selection test p-value 0.000000, feature rel1_6
59. feature selection test p-value 0.000000, feature rel1_5

```
60. feature selection test p-value 0.000000, feature rel1_4
61. feature selection test p-value 0.000000, feature procds
62. feature selection test p-value 0.000000, feature nrel
63. feature selection test p-value 0.000000, feature rel3_26
64. feature selection test p-value 0.000000, feature rel3_25
65. feature selection test p-value 0.000184, feature rel3_21
66. feature selection test p-value 0.412632, feature dtype
67. feature selection test p-value 1.000000, feature tier
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/data/classification/tier2/tier2

Feature ranking by random forest classifier:
1. importance 0.174095, feature relt_2
2. importance 0.161364, feature relt_0
3. importance 0.095066, feature rel3_5
4. importance 0.081929, feature rel3_9
5. importance 0.078290, feature rel1_7
6. importance 0.076775, feature relt_1
7. importance 0.074796, feature rel1_3
8. importance 0.074729, feature rel2_6
9. importance 0.017962, feature proc_evts
10. importance 0.011349, feature rel3_0
11. importance 0.009423, feature rel3_8
12. importance 0.008574, feature cpu
13. importance 0.008404, feature size
14. importance 0.007475, feature rel3_11
15. importance 0.007321, feature rel3_3
16. importance 0.006788, feature rel1_6
17. importance 0.006681, feature nfiles
18. importance 0.006606, feature rel2_1
19. importance 0.005819, feature rel2_0
20. importance 0.005471, feature rel1_4
21. importance 0.005434, feature rel1_5
22. importance 0.005316, feature nlumis
23. importance 0.004555, feature rel3_7
24. importance 0.004482, feature rel2_3
25. importance 0.004441, feature rel2_4
26. importance 0.004290, feature nblk
27. importance 0.004144, feature rel3_2
```

28. importance 0.003561, feature rel3_1
29. importance 0.003465, feature nevt
30. importance 0.003312, feature rel3_10
31. importance 0.003310, feature primds
32. importance 0.003176, feature rel2_2
33. importance 0.003116, feature rel3_4
34. importance 0.003079, feature rel3_6
35. importance 0.002615, feature parent
36. importance 0.002448, feature dataset
37. importance 0.002403, feature nrel
38. importance 0.002395, feature era
39. importance 0.002317, feature creator
40. importance 0.002023, feature rel1_2
41. importance 0.001595, feature dbs
42. importance 0.001516, feature procds
43. importance 0.001475, feature rel3_15
44. importance 0.001305, feature rel1_0
45. importance 0.001164, feature rel3_13
46. importance 0.001032, feature rel2_5
47. importance 0.000895, feature rel3_17
48. importance 0.000710, feature rel3_12
49. importance 0.000448, feature rel3_16
50. importance 0.000369, feature rel3_14
51. importance 0.000352, feature rel2_9
52. importance 0.000216, feature rel2_8
53. importance 0.000074, feature rel2_11
54. importance 0.000055, feature rel3_23
55. importance 0.000000, feature rel1_1
56. importance 0.000000, feature rel2_10
57. importance 0.000000, feature rel2_7
58. importance 0.000000, feature rel3_18
59. importance 0.000000, feature rel3_19
60. importance 0.000000, feature rel3_20
61. importance 0.000000, feature rel3_21
62. importance 0.000000, feature rel3_22
63. importance 0.000000, feature dtype
64. importance 0.000000, feature rel3_24
65. importance 0.000000, feature rel3_25

```
66. importance 0.000000, feature rel3_26
67. importance 0.000000, feature tier

[tili@lxplus0079 testset]$ check_prediction --fin="$datasetdir"/testset/transform/
> cy,precision,recall,f1
Score metric (accuracy_score): 1.0
Score metric (precision_score): 1.0
Score metric (recall_score): 1.0
Score metric (f1_score): 1.0
```

Appendix C. Output when using mostly future conference counts as features

```
[tili@lxplus0079 testset]$ model --learner=RandomForestClassifier --idcol=id --tar
> ler --newdata=''$datasetdir''/testset/transform_conf/dataframe-20140507-20140513
> acy,precision,recall,f1

RandomForestClassifier(bootstrap=True, compute_importances=None,
criterion='gini', max_depth=None, max_features='auto',
max_leaf_nodes=None, min_density=None, min_samples_leaf=1,
min_samples_split=2, n_estimators=10, n_jobs=1,
oob_score=False, random_state=123, verbose=0)
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/data/classification/tier2/merge
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/DCAFPIlot/slc6_amd64_gcc481/ext
fprob is deprecated in scipy 0.14, use stats.f.sf or special.fdtrc instead

warnings.warn(depdoc, DeprecationWarning)

Feature ranking by ANOVA F test:
1. feature selection test p-value 0.000000, feature dataset
2. feature selection test p-value 0.000000, feature dbs
3. feature selection test p-value 0.000000, feature 60wk
4. feature selection test p-value 0.000000, feature 65wk
5. feature selection test p-value 0.000000, feature 70wk
6. feature selection test p-value 0.000000, feature 55wk
7. feature selection test p-value 0.000001, feature 50wk
8. feature selection test p-value 0.001715, feature 30wk
9. feature selection test p-value 0.004668, feature 35wk
```

```
10. feature selection test p-value 0.033383, feature 45wk
11. feature selection test p-value 0.309144, feature 10wk
12. feature selection test p-value 0.431895, feature 0wk
13. feature selection test p-value 0.480560, feature 1wk
14. feature selection test p-value 0.508494, feature 25wk
15. feature selection test p-value 0.580045, feature 4wk
16. feature selection test p-value 0.849305, feature 20wk
17. feature selection test p-value 0.859428, feature 15wk
18. feature selection test p-value 0.945397, feature 2wk
19. feature selection test p-value 1.000000, feature 6wk
20. feature selection test p-value 1.000000, feature 40wk

Feature ranking by Chi Squared test:
1. feature selection test p-value 0.000000, feature dataset
2. feature selection test p-value 0.000000, feature dbs
3. feature selection test p-value 0.000000, feature 70wk
4. feature selection test p-value 0.000000, feature 65wk
5. feature selection test p-value 0.000000, feature 60wk
6. feature selection test p-value 0.000000, feature 55wk
7. feature selection test p-value 0.000000, feature 30wk
8. feature selection test p-value 0.000000, feature 50wk
9. feature selection test p-value 0.000000, feature 35wk
10. feature selection test p-value 0.000000, feature 10wk
11. feature selection test p-value 0.000120, feature 45wk
12. feature selection test p-value 0.000192, feature 0wk
13. feature selection test p-value 0.000801, feature 1wk
14. feature selection test p-value 0.004897, feature 4wk
15. feature selection test p-value 0.008272, feature 25wk
16. feature selection test p-value 0.381273, feature 20wk
17. feature selection test p-value 0.385936, feature 15wk
18. feature selection test p-value 0.408017, feature 40wk
19. feature selection test p-value 0.736402, feature 2wk
20. feature selection test p-value 0.966743, feature 6wk
/afs/cern.ch/user/t/tili/mywork/DCAFPilotCalMining/data/classification/tier2/merge

Feature ranking by random forest classifier:
1. importance 0.724537, feature dataset
2. importance 0.178246, feature dbs
```

```
3. importance 0.008233, feature 50wk
4. importance 0.007089, feature 65wk
5. importance 0.006662, feature 70wk
6. importance 0.006617, feature 55wk
7. importance 0.006465, feature 45wk
8. importance 0.006123, feature 0wk
9. importance 0.005947, feature 2wk
10. importance 0.005602, feature 60wk
11. importance 0.005536, feature 1wk
12. importance 0.005083, feature 4wk
13. importance 0.005015, feature 6wk
14. importance 0.004994, feature 35wk
15. importance 0.004759, feature 15wk
16. importance 0.004361, feature 30wk
17. importance 0.004266, feature 20wk
18. importance 0.004029, feature 25wk
19. importance 0.003225, feature 10wk
20. importance 0.003212, feature 40wk

[tili@lxplus0079 testset]$ check_prediction --fin=''$datasetdir''/testset/transfor
> rer=accuracy,precision,recall,f1
Score metric (accuracy_score): 0.960264900662
Score metric (precision_score): 0.777777777778
Score metric (recall_score): 0.636363636364
Score metric (f1_score): 0.7
```

References

[1] V. Kuznetsov, Dcaf pilot for cms: pilot project for cms computing data-mining, 2015. URL: http://www.lepp.cornell.edu/~vk/Talks/Pilot1/it.html.

[2] T. Li, Dcaf pilot calendar mining site, 2015. URL: https://github.com/tli12/DCAFPilotCalMining.

[3] V. Kuznetsov, Dmwm analytics project site, 2015. URL: https://github.com/dmwm/DMWMAnalytics.

[4] S.-L. developers, 1.11.2.5. feature importance evaluation, scikit-learn version 0.16.1 user guide, 2015. URL: http://scikit-learn.org/stable/modules/ensemble.html#feature-importance-evaluation.

[5] S.-L. developers, Categorical split for decision tree discussion, 2015. URL: https://github.com/scikit-learn/scikit-learn/pull/3346.