Prediction of primary sites of cancer using random forest classification

Tobias Mai
12182767

CMSC 33750
University of Chicago

October 25, 2017

1 Data preparation

1.1 Setting up the initial data frame

The first step after `curl`-ing the JSON data was to crop the additional meta-information given in the newly created file. The structure of this file is depicted in figure 1. Everything except the array contained within `json["data"]["hits"]` was dropped.

```
{
        ”data”: {
                ”pagination”: {
                        ”total”: 182972,
                        ”count”: 182972,
                        ”size”: 200000,
                        ”sort”: ””,
                        ”from”: 0,
                        ”page”: 1,
                        ”pages”: 1
                },
                ”hits”: [...]} ,
        ”warnings”: {}
}
```

Figure 1: Structure of the `curl`-ed JSON-File

To analyze the contents of this array, it was converted to a data frame using the python library *pandas*[1]. The data frame's summary is presented in figure 2

```
   Data columns (total 11 columns):
case.case_id                 182972 non-null
case.primary_site            182972 non-null
case.project.project_id      182972 non-null
id                           182972 non-null
ssm.chromosome               182972 non-null
ssm.consequence              182972 non-null
ssm.end_position             182972 non-null
ssm.genomic_dna_change       182972 non-null
ssm.mutation_subtype         182972 non-null
ssm.mutation_type            182972 non-null
ssm.start_position           182972 non-null
         memory usage: 15.4+ MB
```

(a) summary of the data frame

```
             ssm.consequence
0  [’transcript’: ’gene’: ’symbol’: ’EFR3A’,...]
1  [’transcript’: ’gene’: ’symbol’: ’UBAP2’,...]
2  [’transcript’: ’gene’: ’symbol’: ’CPS1’,...]
3  [’transcript’: ’gene’: ’symbol’: ’CBWD1’,...]
4  [’transcript’: ’gene’: ’symbol’: ’GALNT13’,...]
```

(b) list in ssm.consequence

Figure 2: Analysis of the data

[1] `http://pandas.pydata.org/`

Comparing figure 2a to figure 1, one can already identify that there are no issues related to missing values. However, since *pandas* cannot deal with lists within the json structure, some additional manual processing was required to retrieve the gene information from the lists still contained in the column labeled ”ssm.consequence” (see figure 2b. Since this list basically contained multiple entries of the same particular gene, this information was extracted. Thus the list was converted to a single gene value.

Further, the following columns were dropped since they either contained a single value across all records or their information was not needed anymore:

- case.project.project_id
- ssm.end_position and ssm.start_position (redundant since the gene information is given)
- ssm.mutation_type and ssm.mutation_subtype (only one value since the data was explicitly filtered for these values)

1.2 Grouping the data by case

The records currently given resemble single mutations observed within a particular case. However, to infer the primary site of a cancer type by case, this data has to be *grouped by* case.case_id. One major issue of doing this with the current dataset is that we did not select an features yet, thus we don't know *what* to group and even more important, *how* to group. For example, grouping over a column like ”ssm.consequences”, which holds the gene names affected by the mutation, is not trivial, canonical aggregation functions like *sum* or *average* are not applicable to string values. Therefore, the following section deals with how the features were selected and how they were transformed into numerical values.

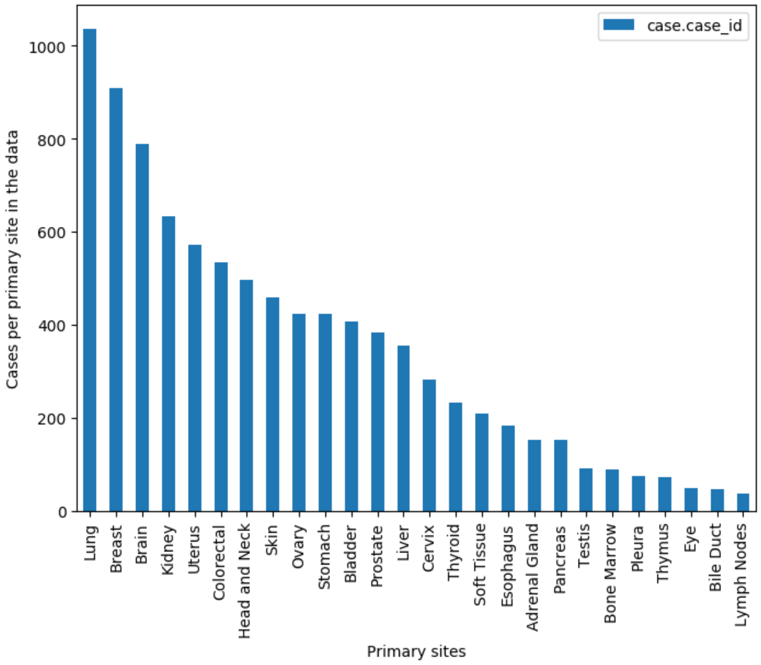


Figure 3: Cases per primary site of cancer

Although the data was not really grouped yet, it is possible to analyze how many cases in the given data belong to a specific type of cancer. This is presented in figure 3.It is easy to identify that the data is greatly unbalanced. On the one hand, there are over 1000 cases of Lung Cancer, on the other one, only 37 belong to Lymph nodes as primary site.

2 Feature selection

2.1 Features directly taken from the data

Two basic types of genomic information can be directly retrieved from the data:

- "ssm.chromosome": the chromosome affected by the mutation
- "ssm.consequence": the gene affected by the mutation

In addition, the column "ssm.genomic_dna_change" contains a string comprising chromosome, gene locus and the specific somatic mutation each gene had undergone in particular. By utilizing trivial means of string parsing/slicing, a new column, "ssm.genomic_dna_change.nucleotide_change" can be created that contains the latter value of the string. In turn, the information contained in the column "ssm.genomic_dna_change" has now become redundant and can be dropped.

However, as previously mentioned in section 1.2, these string values have to be transformed into numerical values to enable effective aggregation. In general, the idea was to at least have one *binary* column/feature for the cartesian product of each chromosome, gene and somatic mutation present in the data. This would amount to $n_{chromosome} * n_{gene} * n_{mutation}$ features being used. Accounting for the fact that the data comprises

- 24 different chromosomes,
- 18177 different genes, and
- 12 somatic mutations,

we would end up at more than **5 million** features being used for the classifier. Considering both the negative impact of *high-dimensional feature-space* (also referred to as *feature sparsity*) in general and limitations imposed by *available CPU and RAM*, this endeavor should be greatly discouraged. Therefore, the following features will be used instead:

- cartesian product of chromosomes and somatic mutations: 288 columns
- genes: 18177 columns

Without incorporating any further knowledge from cancer biology, creating the crossover of chromosomes and somatic mutations was an attempt to capture the assumption of a correlation, even without the gene being explicitly given. The genes were taken individually to avoid further inflation of the feature space. Table 4 presents how the data could look like after the feature columns have been created. In this simplified example, the mutation with ID 0 affected gene *APC* at chromosome 1, and the specific somatic mutation was C>T. The mutation with ID 1 in turn affected gene *NF1* at chromosome 5, and the specific somatic mutation was C>T, too.

ID	APC	NF1	chr5 + C>T	chr1 + C>T
0	1	0	0	1
1	0	1	1	0

Figure 4: Exemplary data in a simplified version of the feature frame

Seizing upon the notion already mentioned in section 1.2, the created records can now be grouped by case.case_id. As aggregation function, sum was simply used. The idea was that different values in the occurrences of e.g. particular genes being affected might be an indicator for different types of cancer.

Unfortunately, it turned out that using a *pandas* data frame was inefficient in terms of setting up the required 18,465 columns for the chosen features. Therefore, grouping would in practice not be as easy as calling the *groupby* function provided by *pandas*. However, *numpy*[2] arrays did prove to be more efficient for various reasons[3]. In the end, a manually written equivalent of the *groupby* function was used to benefit from using *numpy*.

2.2 Additional features

In addition to these features taken directly from the data, the *total amount of mutations* observed for a particular case was computed, i.e. how many of the initial records would belong to a specific case_id. This should reflect different levels of aggressiveness, which could further help to distinguish different types of cancer.

3 Split into train and test data

A split into 90% and 10% for training respectively test data was chosen for any subsequent models to be built and evaluated. Further, for the second round of random forest mentioned in section 5.2, a 10-fold cross-validation was used to ensure that every single case was tested once.

4 Classifier: random forest

4.1 Structure of the approach

Random forest was chosen to be used throughout the entirety this classification problem. One advantage of random forest is the ability to rank the features according to their importance towards the final prediction. Therefore, an iterative approach will be used, making use of two consecutively executed classifiers:

1. Random forest to determine the accuracy when using all of the features discussed in section 2

2. Consecutive random forest classifier run with a reduced amount of features, i.e. incorporating the knowledge about feature importance gained from the first classifier

[2] `http://www.numpy.org/`

[3] less overhead, ability to adjust costly 64 bit integer fields to int8, resp. boolean

5 Classifier Evaluation

5.1 First run

9006 cases' primary sites and the corresponding 9006x18465 feature matrix were fed into *scikit-learn's RandomForestClassifier*[4]. The remaining 91 labels (10%) and their corresponding features were held back for testing purposes. 500 classification trees were used, and no maximal required impurity decrease was specified. One could argue that this may lead to overfitting, i.e. the classifier perfectly learning the training data instead of the structure behind the data. However, this overfitting can be assumed to be less severe in comparison to using a single decision tree.

The resulting model was able to correctly classify 30.45% of the test data. This might be nowhere near to an accurate prediction to be used in practice, however it still outperforms the *random choice*, which would be $\frac{1}{26}$[5]. The confusion matrix of the forest is given in figure 5

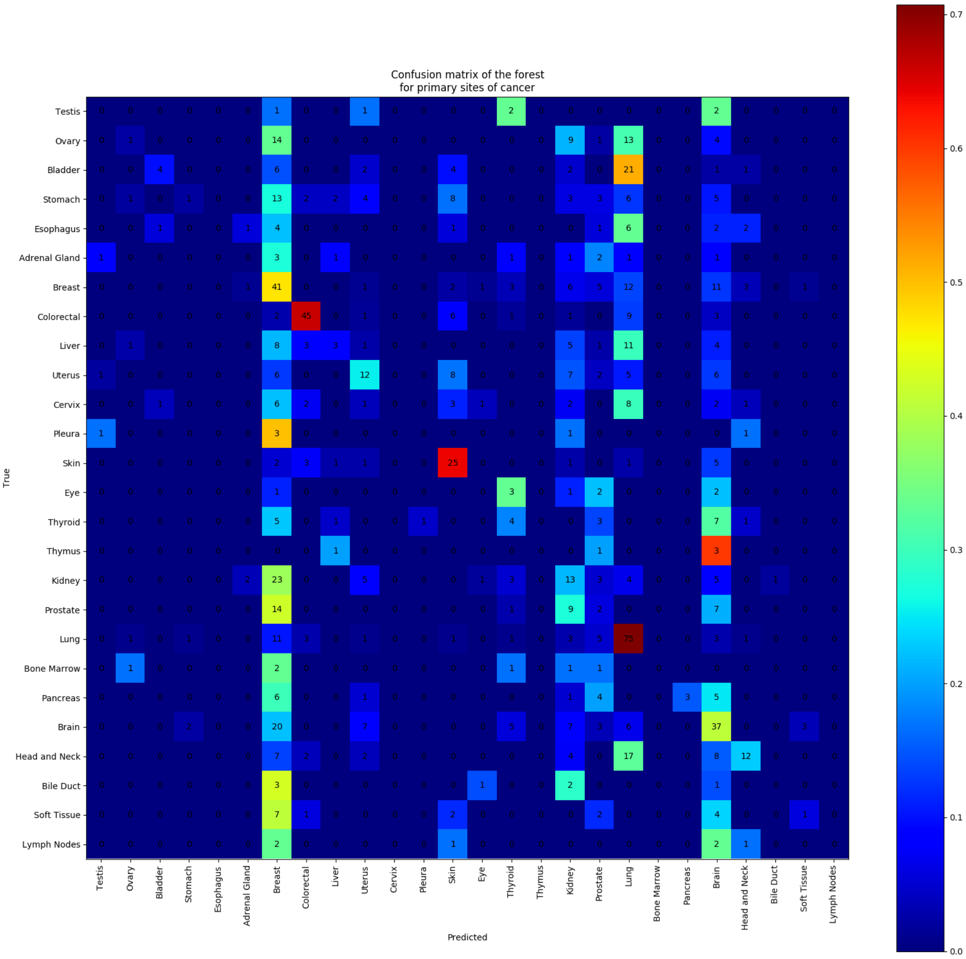


Figure 5: Confusion matrix of the first random forest classifier

[4] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5] since there were 26 different labels to choose from

First of all, in case of a good classification, one should expect the cells running diagonally downward from the left upper corner to be yellow to red shaded. This is clearly not the case in the given matrix. The only acceptable true positives were achieved for lung, colorectal, and skin. However, inspecting the scale, even the correct identification of lung cancer comes with only 70% accuracy.

In addition, the classifier's three most frequent predictions irrespective of real type of cancer were breast, lung and brain. This leads to both high false positive and high false negative classifications for basically each type of cancer except for the three aforementioned ones. By comparing this result to the abundance of each cancer type in the data (see figure 3), one could easily conclude that the class imbalance is a main driver for this phenomenon, because lung, breast, and brain are the top three most frequent labels in the data. Therefore, the class imbalance might have been a major influence to the fairly low prediction accuracy.

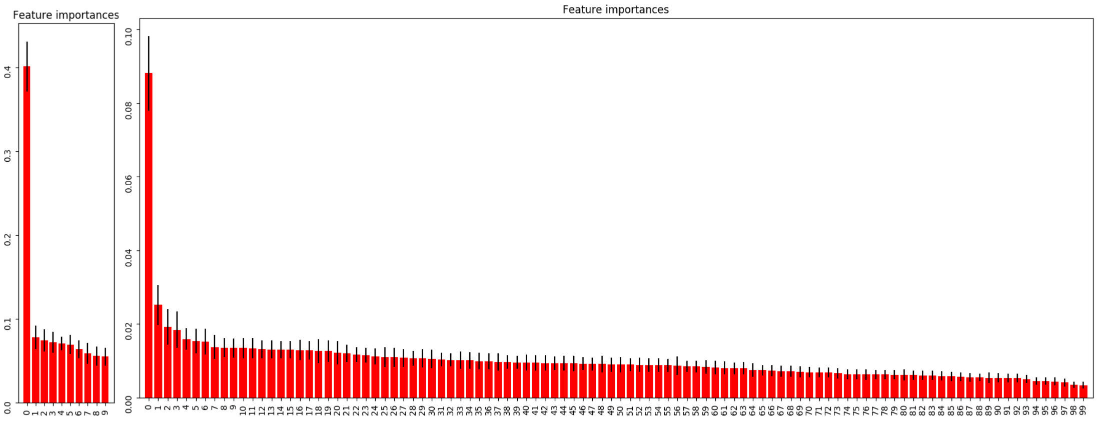
5.2 Second run

As previously mentioned in section 4.1, the second run aimed at rerunning the classification, however this time with a decreased amount of features. In addition, different amounts of features shall be evaluated, ranging from 1 to 100 in increments of 10. Therefore, the 100 most influential features as determined by the first random forest classifier, are listed in figure 6

0: MutationCount	1: APC	2: chr5+C>T	3: chr1+C>T
4: chr1+G>A	5: chr1+C>A	6: chr2+C>T	7: chr17+G>A
8: chr1+G>T	9: chr2+G>A	10: chr19+G>A	11: chr3+C>T
12: chr2+C>A	13: chr17+C>T	14: ATRX	15: TP53
16: chr19+C>T	17: chr3+G>A	18: chr12+C>T	19: chr12+G>A
20: chrX+G>T	21: chr2+G>T	22: chr6+C>T	23: chr7+G>A
24: chr11+C>T	25: chr6+G>A	26: chr11+G>A	27: chr4+G>A
28: chr5+G>T	29: chr11+C>A	30: chr19+G>T	31: chrX+C>T
32: chr3+C>A	33: chr12+C>A	34: chr7+C>T	35: chr19+C>A
36: chr11+G>T	37: chr8+G>A	38: chrX+G>A	39: chrX+C>A
40: chr17+C>A	41: chr5+G>A	42: chr4+C>T	43: chr7+G>T
44: chr10+C>T	45: chr10+G>A	46: chr6+C>A	47: chr16+G>A
48: chr3+G>T	49: chr5+C>A	50: chr9+G>A	51: chr8+C>A
52: chr7+C>A	53: chr16+C>T	54: chr8+C>T	55: VHL
56: PBRM1	57: chr17+G>T	58: chr9+C>T	59: chr4+C>A
60: chr12+G>T	61: chr6+G>T	62: PTEN	63: chr15+G>A
64: chr4+G>T	65: chr14+C>T	66: chr15+C>T	67: ARID1A
68: chr20+C>T	69: chr8+G>T	70: chr14+G>A	71: chr16+G>T
72: CDKN2A	73: chr10+G>T	74: chr10+C>A	75: CDH1
76: chr20+G>A	77: chr1+C>G	78: chr13+C>T	79: chr9+C>A
80: chr16+C>A	81: chr1+G>C	82: chr9+G>T	83: chr14+C>A
84: chr1+T>A	85: TTN	86: chr15+C>A	87: chr18+C>T
88: chr22+C>T	89: chr14+G>T	90: KMT2D	91: chr22+G>A
92: chr2+T>A	93: chr20+G>T	94: chr1+A>T	95: chr15+G>T
96: NF1	97: chr2+C>G	98: chr13+G>A	99: chr2+G>C

Figure 6: 100 most influential features determined by first forest classifier

First of all, it is quite astonishing that the overall mutation count, mentioned in section 2.2, was rated to be the most influential feature. This could indicate that, up to a certain extent, the different cancer types are distinguishable by their ”aggressiveness”, i.e. corresponding number of mutations. Second, among the 100 features, only 12 were gene-based. Figure 7 presents the importances for each feature for a second random forest classification with either 10 or 100 of the most influential features used.



(a) 10

(b) 100

Figure 7: Feature importance when choosing the 10 and 100 most influential features

Again, in both plots, the importance of the overall mutation count feature (index 0) is standing out. Further, the overall shape of the importance bar plots indicates that already the 90 features added from 7a to 7b are of a dwindling importance to the overall prediction quality. This claim can be further supported by comparing the different achieved prediction rates by each of the second random forest runs, depicted in figure 8.

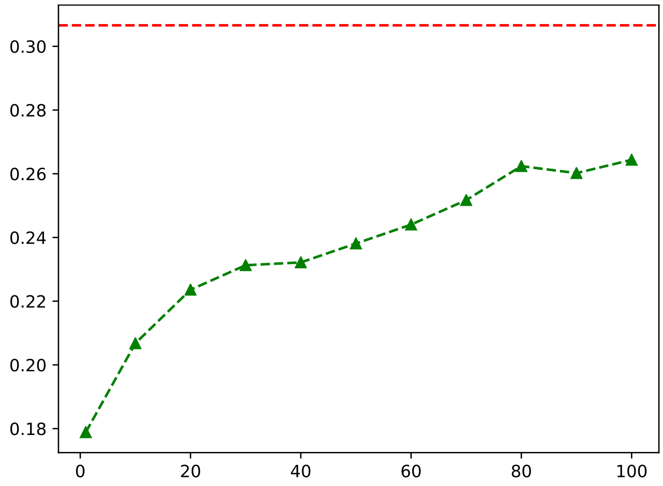


Figure 8: Confidence levels of second random forest runs

The percentages given in this graph were obtained by running 10-fold cross-validation for each given number of features respectively, and averaging over the obtained accuracies. The red dashed line indicates the 30.45% prediction accuracy of the first run. Again, it is astonishing that the overall mutation count already accounts for more than 50% of the maximally achieved prediction rate of the first run. Further, adding features[6] seems to have a diminishing benefit on the prediction accuracy, so that the curve depicted in figure 8 assumes a logarithmic shape.

6 Classifier improvement

Overall, it has to be emphasized that 30% prediction accuracy is nowhere close to a possible application of this classification method in practice. Notwithstanding, 30% still beats the random choice, indicating that the features chosen capture a correlation between them and the primary site of cancer up to a certain extent. One improvement that seems to be beneficial would be to balance the classes before running the classifier. The confusion matrix has particularly presented that the prediction is correlated with the abundance of a class within the training data.

Furthermore, not all possible feature combinations have been tested. Especially, the Cartesian product of chromosomes, genes, and somatic mutations rendered to be infeasible concerning limitations in the given computational power of a personal computer. Also, combining chromosomes and somatic mutations was not proven to be beneficial. For example, a run taking only somatic mutations, irrespective of gene or chromosome, into account, might show to be better in terms of prediction accuracy.

[6] meaning adding features out of the first run's feature pool