**DNA methylation**

**Creating a proper dataset**

**01.**

We downloaded 3 datasets for meth type from TCGA.

1. KIRC\_HumanMethylation450.data 🡪 Clear cell
2. KIRP\_HumanMethylation450.data 🡪 Papilary cell
3. KICH\_HumanMethylation450.data 🡪 Chromophobe

**02.**

In the datasets, columns contained patients and rows contained the protein types (features). Since in TCGA, they divided the datasets for each subtype, in each dataset there were not a target row. So, we added a target row as ‘Subtype’ for each dataset with some integer values. Values can define as.

* ‘0’: KIRC
* ‘1’: KIRP
* ‘2’: KICH

Meth data procedure is littlebit different from other omic types since these datasets were very larger. So this implementation was done in 3 ways as;

1. Considering whole dataset (whole features)

* Here we did not do any feature selection. Just we tried how the whole dataset perform with our procedure.

1. Feature selection for select limited number of features

* Here we selected 200 best features using mutual information and created a small dataset.

1. Considering new dataset with selected features

* In this step we used the dataset with selected features (200) and tested the performance.

**1. Considering whole dataset (whole features)**

**Dataset initialization**

**01.**

Here we started to create our models in python 3 environment. Here we imported all the datasets.

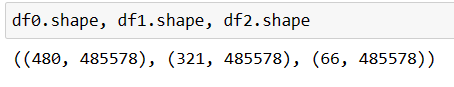


Reason for using set\_index is, if we did not define the index, by default Pandas will create an index for the DataFrame. It makes things a little more confusing, because by default the “index” is just the range of numbers starting at 0.

Here we swap the rows and columns using *transpose()* method to get the features and label into columns and patients into rows.

**02.**

Here are the shapes of each dataset.



We can clearly see that the number of features is are equal.

**Data Preprocessing**

**1. Data Integration**

**01.**

Now we need to combine these 3 datasets. We used *pandas.concat* method for combine the datasets.



**02.**

Here is the shape of new data frame (Meth data).

* **(867, 485578)**

Here showing **867** records. It just the combination of df0, df1 and df2 (480+321+66=867).

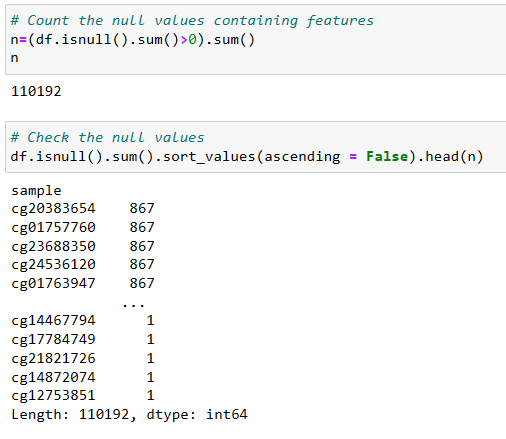
Here showing **485578** columns as in each data frame (each subtype).

**2. Data Cleaning**

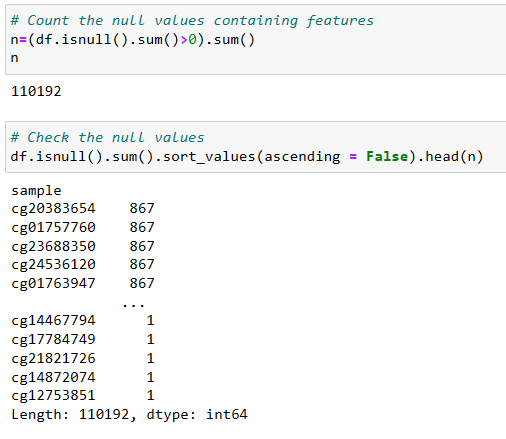
**01.**

Here we check for missing values.

By this visualization it is difficult to take a decision about null values. So, we count the features which containing null values and try to visualize them.



As you can see, we found that there 110192 features contain at least one null value. And we try to see those 110192 features and their null value count.



**03.**

Since we had altogether 485578 features, we decided to remove those 110192 null values containing features. Then we got 375386 usable features (+ labels) up to now.

**04.**

Then we count the zero values in the features. We observed that there are no e zero values in this dataset.

**3. Dataset balancing**

**01.**

Then we viewed the value count of each subtype.

* KIRC - 480
* KIRP - 321
* KICH - 66

As we can see the dataset is unbalance over each subtype.

**02.**

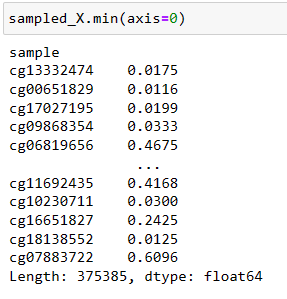
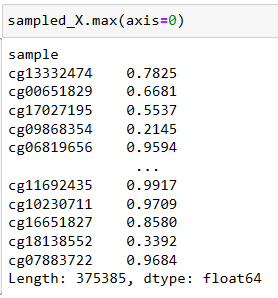
We handle the unbalancing of recodes using up sampling and under sampling to get a fair and balanced dataset.

* KIRC - 100
* KIRP - 100
* KICH - 100

So, we get a balanced dataset with 300 records.

**4. Feature Scaling**

Here I checked whether the dataset need a scaling. For that I just print the max and min values of each feature.



As we can see the data set has different but small range of values.

**Test SVM**

**01.**

First, we try to find the best kernel to use for further process. Here we used cross validation for this test.

* Accuracy SVM with rbf kernel: 0.87 (+/- 0.11)
* Accuracy SVM with poly kernel: 0.92 (+/- 0.07)
* **Accuracy SVM with linear kernel: 0.96 (+/- 0.06)**
* Accuracy SVM with sigmoid kernel: 0.65 (+/- 0.06)

**linear** kernel performed well.

**02.**

Then, we find the accuracy without data transformation. Then we tried both standardization and normalization. So, we observed that here we do not need any transformation.

* **Accuracy SVM without transformation: 0.96 (+/- 0.06)**
* Accuracy SVM with Standardization: 0.96 (+/- 0.06)
* Accuracy SVM with Normalization: 0.96 (+/- 0.07)

**Test RF**

**01.**

First, we try different combinations of parameters with the help of GridSearchCV. Here we tested best 3 parameters in RF and took the best combination.

* Best parameters are:
  + 'criterion': entropy,
  + 'max\_depth': 8,
  + 'n\_estimators': 500

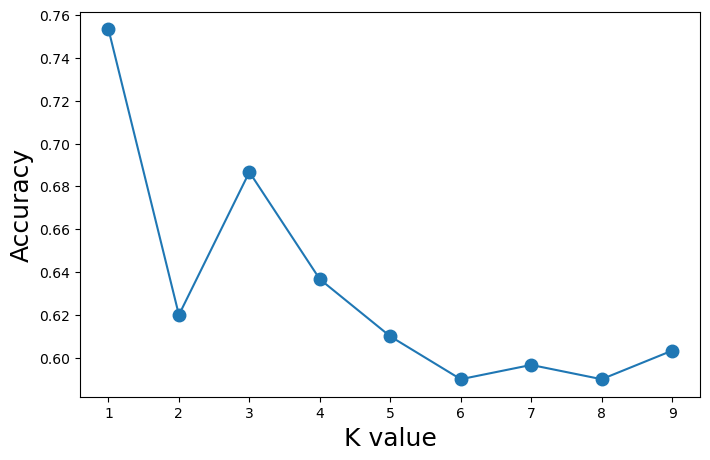
Then we did the cross validation.

* Accuracy RF with best\_params: 0.95 (+/- 0.04)

**Test KNN**

**01.**

First, we try to find the best k value to use for further process. Here accuracy is got from cross validation’s (cv=5) mean value.



Here k=3 is showing the best accuracy. But since we have 3 classes, we choose **4** as the optimal value for k value.

**02.**

Then, we find the accuracy without data transformation. After that we tried both standardization and normalization. Here normalization performed well with KNN here. Here also for testing we used cross validation for this selection.

* Accuracy KNN without transformation: 0.64 (+/- 0.10)
* Accuracy KNN with Standardization: 0.69 (+/- 0.04)
* **Accuracy KNN with Normalization: 0.71 (+/- 0.03)**

**2. Feature selection for select limited number of features**

**Dataset initialization**

**01.**

Here we started to create our models in python 3 environment. Here we imported all the datasets.

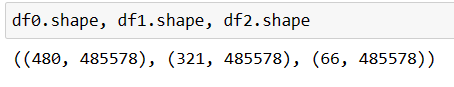


Reason for using set\_index is, if we did not define the index, by default Pandas will create an index for the DataFrame. It makes things a little more confusing, because by default the “index” is just the range of numbers starting at 0.

Here we swap the rows and columns using *transpose()* method to get the features and label into columns and patients into rows.

**02.**

Here are the shapes of each dataset.



We can clearly see that the number of features is not equal.

**Data Preprocessing**

**1. Data Integration**

**01.**

Now we need to combine these 3 datasets. We used *pandas.concat* method for combine the datasets.



**02.**

Here is the shape of new data frame (Meth data).

* **(867, 485578)**

Here showing **867** records. It just the combination of df0, df1 and df2 (480+321+66=867).

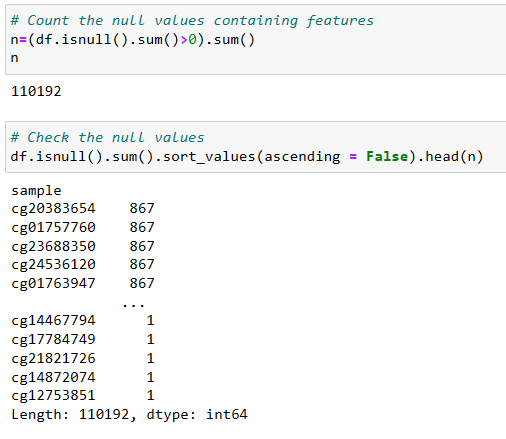
Here showing **485578** columns as in each data frame (each subtype).

**2. Data Cleaning**

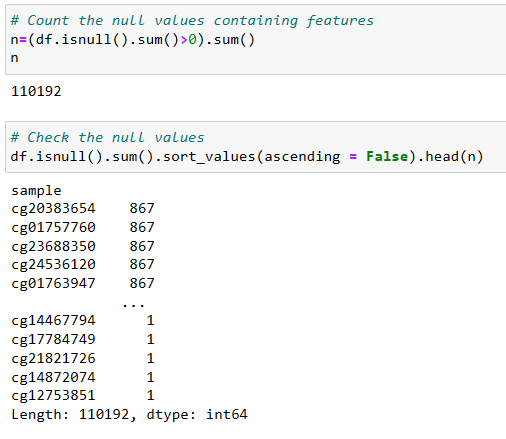
**01.**

Here we check for missing values.

By this visualization it is difficult to take a decision about null values. So, we count the features which containing null values and try to visualize them.



As you can see, we found that there 110192 features contain at least one null value. And we try to see those 110192 features and their null value count.



**03.**

Since we had altogether 485578 features, we decided to remove those 110192 null values containing features. Then we got 375386 usable features (+ labels) up to now.

**04.**

Then we count the zero values in the features. We observed that there are no e zero values in this dataset.

**3. Feature Selection**

Here we tried feature selection using mutual information. So, we created a csv file with 200 best features for further process. Reason for this step is Pearson correlation was failed with our initial large dataset. It approximate required 1TB memory. So we decided to go with mutual information first and create a dataset that we can handle easily.

**3. Considering new dataset with selected features (200)**

**Dataset initialization**

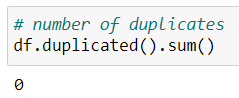
Here we imported dataset which was feature selected in feature selection step. Initial shape of that dataset is (867, 202). 202 columns came from 200 features, label (subtype) and index patient ID. As patient ID can be taken as the index, as our first step we make it as the index.

**Data Preprocessing**

**Data Cleaning**

**01.**

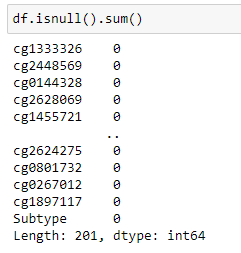
First, we try to drop duplicate records.



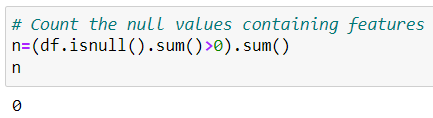
But we found there weren’t any duplicates.

**02.**

Then we check for missing values.



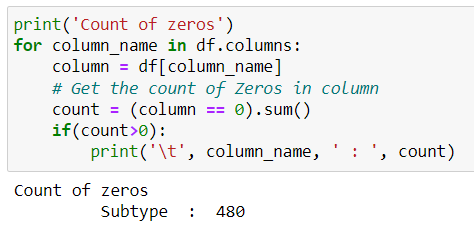
It seems like there are no null values. To ensure that, we can take the sum of null values.



There are no null values in this dataset. So doesn’t need to handle null values.

**03.**

Then we count the features which have zero values.

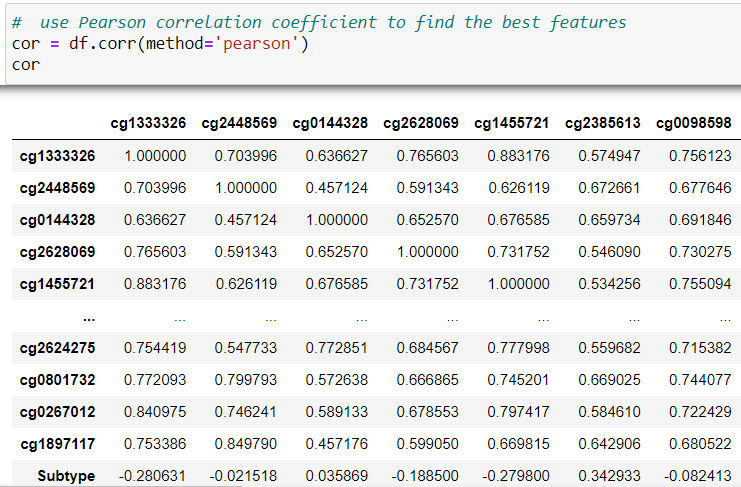


We only found zeros in Subtype column. But we don’t need to do anything for that. Because Subtype is the label column of this dataset and 0 identifies a defined class value.

**Feature Selection - Pearson correlation coefficient**

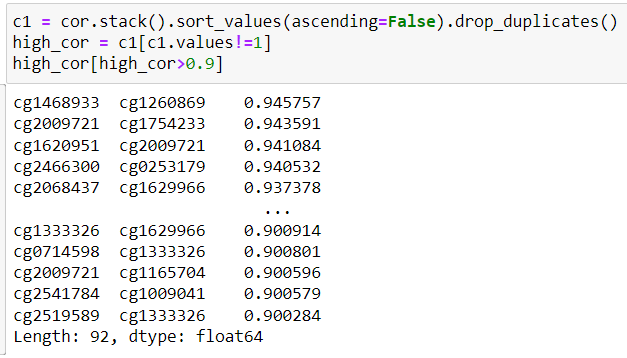
**01.**

Getting the correlation coefficients in dataset

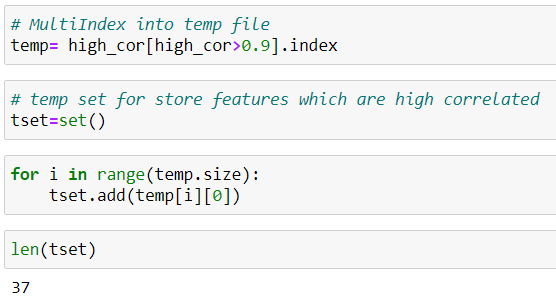


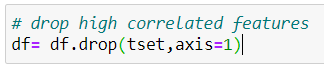
**02.**

In this step we removed the features which overly corelated. For that we taken a threshold value of 0.90 and removed relevant features.



Since these values are high, it means these are somewhat linearly dependent with other features. hence those effect on the dependent variable is almost same. So we can drop one of them.





Removed feature(37):

'cg0093751',

'cg0095282',

'cg0103341',

'cg0115714',

'cg0195430',

'cg0663891',

'cg0666533',

'cg0714598',

'cg0733846',

'cg0834842',

'cg0859706',

'cg1009041',

'cg1165704',

'cg1182264',

'cg1333326',

'cg1345860',

'cg1407833',

'cg1468933',

'cg1519962',

'cg1561875',

'cg1612242',

'cg1620951',

'cg1629966',

'cg1715891',

'cg1754233',

'cg1790158',

'cg1897117',

'cg2009721',

'cg2068437',

'cg2181533',

'cg2233076',

'cg2362287',

'cg2391829',

'cg2466300',

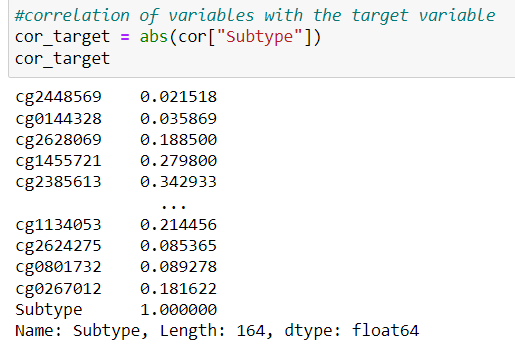
'cg2486424',

'cg2519589',

'cg2541784'

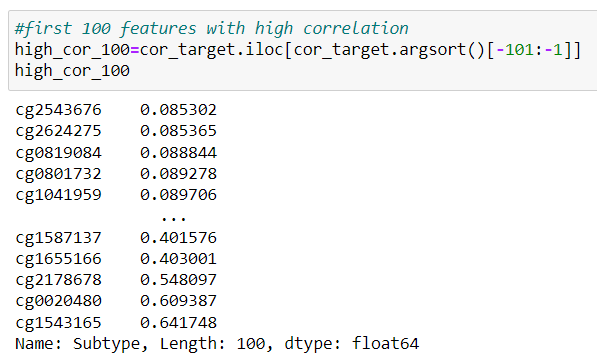
**03.**

Taking the absolute correlation of features with the target variable.



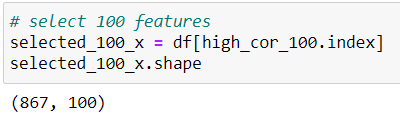
**04.**

Initially We got the best 100 features for further process.



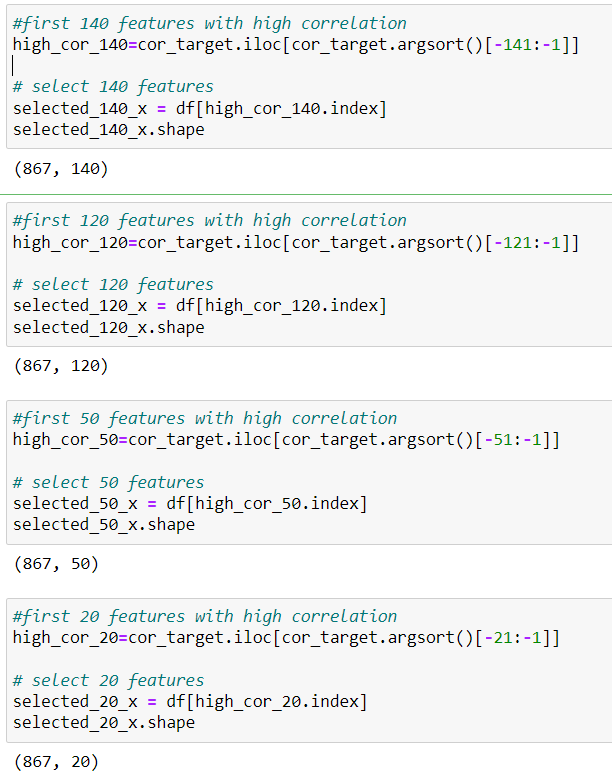
Here features are taken in ascending order of coefficient. So, we can see that it is in the range of 0.085302 to 0.641748. Actually, this is a regular correlation range.

Selected 100 features



**05.**

Likewise in the step 04, We build models with getting different feature amounts (20,50).

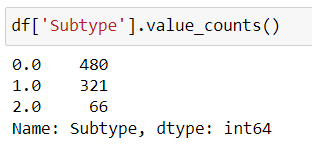


**Dataset balancing**

**01.**

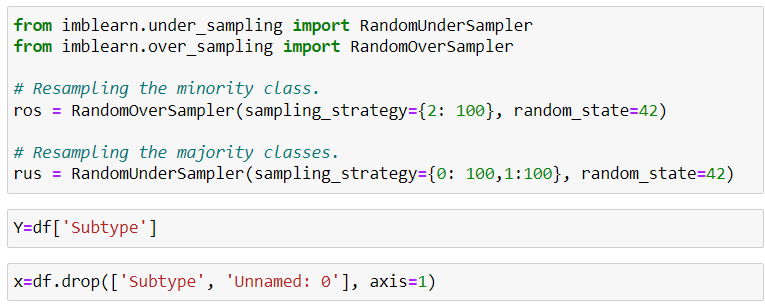
Then we viewed the value count of each subtype.

* Initial feature count from each class (Before sampling)



As it shows the data frame is highly imbalanced. So, I decided to handle these imbalancing. For that I used both Oversampling and Undersampling.

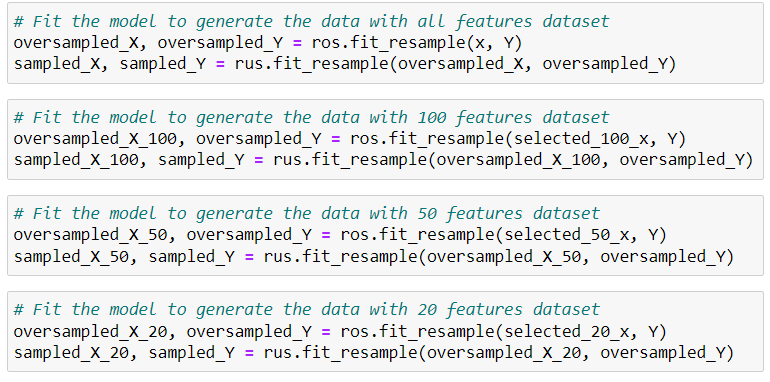
* Initial steps of sampling process



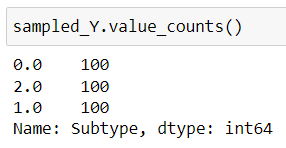


At last, I want to find out the required feature count which gives the best performing model. So, I did the sampling process for all data frames which have different number of features.

* Implement the sampling process for whole data frames

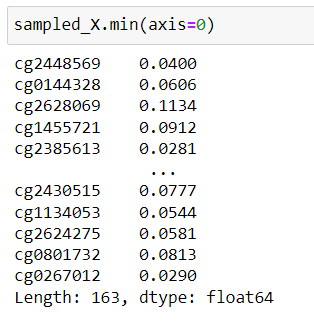
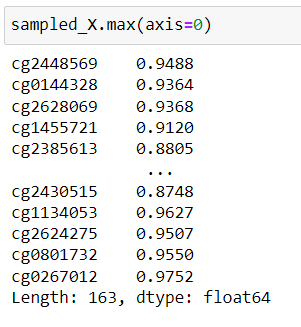


* Derived feature count from each class (After sampling)



**5. Feature Scaling**

Here I checked whether the dataset need a scaling. For that I just print the max and min values of each feature.



As we can see the data set has different but small range of values.

**Test SVM**

**01.**

First, we try to find the best kernel to use for further process. Here we used cross validation for this test.

* Accuracy SVM with rbf kernel: 0.90 (+/- 0.06)
* **Accuracy SVM with poly kernel: 0.92 (+/- 0.08)**
* **Accuracy SVM with linear kernel: 0.92 (+/- 0.08)**
* Accuracy SVM with sigmoid kernel: 0.18 (+/- 0.12)

Both poly and linear kernels performed well. So, we decided to check performances of both kernels.

**02. a.**

Then, we find the accuracy using **linear** kernel without data transformation, standardization and normalization. So, we observed that standardization performed well with SVM. Here also used cross validation for this test.

* Accuracy SVM without transformation: 0.92 (+/- 0.08)
* Accuracy SVM with Standardization: 0.93 (+/- 0.08)
* Accuracy SVM with Normalization: 0.92 (+/- 0.07)

**b.**

Then, we find the accuracy using **poly** kernel without data transformation, standardization and normalization. So, we observed that standardization performed well with SVM. Here also used cross validation for this test.

* Accuracy SVM without transformation: 0.92 (+/- 0.08)
* Accuracy SVM with Standardization: 0.84 (+/- 0.11)
* Accuracy SVM with Normalization: 0.92 (+/- 0.07)

Considering both kernels, linear kernel performs well and we continued with that kernel for future steps.

**03.**

Then we considered other datasets, which we created with different number of features (20,50,100) and evaluate those models using SVM by cross validations.

* Accuracy SVM using best 20 features: 0.91 (+/- 0.07)
* Accuracy SVM using best 50 features: 0.93 (+/- 0.06)
* Accuracy SVM using best 100 features: 0.93 (+/- 0.06)

And also, we checked the performance for different feature amount around 50.

* **Accuracy SVM using best 30 features: 0.93 (+/- 0.06)**
* Accuracy SVM using best 60 features: 0.92 (+/- 0.07)
* Accuracy SVM using best 40 features: 0.93 (+/- 0.07)

**Test RF**

**01.**

First, we try different combinations of parameters with the help of GridSearchCV. Here we tested best 3 parameters in RF and took the best combination.

* Best parameters are:
  + 'criterion': 'entropy'
  + 'max\_depth': 8,
  + 'n\_estimators': 1000

Then we did the cross validation.

**02.**

Then, we find the accuracy without data transformation. After that we tried both standardization and normalization. But we observed all the performance were same. Reason is, **RF does not need a data transformation since it is a Tree-based model and they are not based on the distance where features have an effect on one another**.

* Accuracy RF with complete features: 0.92 (+/- 0.05)

**03.**

Then we test the model for other datasets, which we created with different number of features (20, 50, 100) and evaluate those models’ using RF by cross validations.

* Accuracy RF using best 20 features: 0.92 (+/- 0.07)
* **Accuracy RF using best 50 features: 0.93 (+/- 0.05)**
* **Accuracy RF using best 100 features: 0.93 (+/- 0.05)**

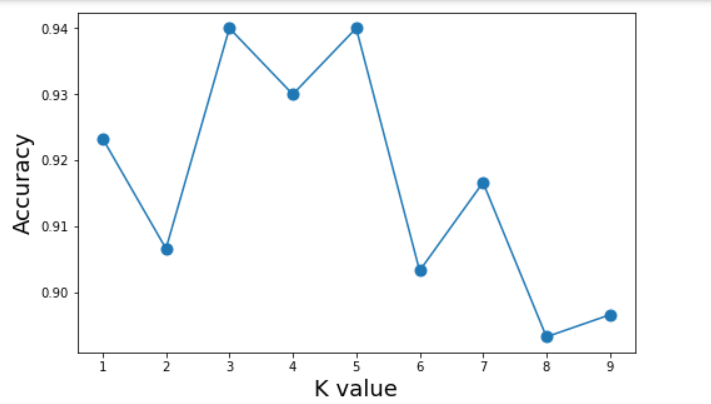
Then we tried with different feature starts from 20

* Accuracy RF using best 40 features: 0.92 (+/- 0.05)
* Accuracy RF using best 60 features: 0.92 (+/- 0.05)
* Accuracy SVM using best 80 features: 0.93 (+/- 0.07)
* Accuracy SVM using best 90 features: 0.93 (+/- 0.04)
* Accuracy SVM using best 110 features: 0.94 (+/- 0.06)
* **Accuracy SVM using best 120 features: 0.94 (+/- 0.05)**
* Accuracy SVM using best 130 features: 0.94 (+/- 0.06)

**Test KNN**

**01.**

First, we try to find the best k value to use for further process. Here accuracy is got from cross validation’s (cv=5) mean value.



Here k=5 is showing the best accuracy. And we have done the remaining process for k=5

**02.**

For k=5; We find the accuracy without data transformation. After that we tried both standardization and normalization. Here all methods performed same with KNN here. Here also for testing we used cross validation for this selection.

* Accuracy KNN without transformation: 0.94 (+/- 0.05)
* Accuracy KNN with Standardization: 0.94 (+/- 0.05)
* Accuracy KNN with Normalization: 0.94 (+/- 0.05)

**03.**

Then we did the standardization for other datasets, which we created with different number of features (20 ,50, 100) and evaluate those models using KNN by cross validations.

* Accuracy KNN using best 20 features: 0.92 (+/- 0.08)
* Accuracy KNN using best 50 features: 0.94 (+/- 0.06)
* Accuracy KNN using best 100 features: 0.93 (+/- 0.06)

Then we tried with different feature amounts between 20 and 100

* Accuracy KNN using best 30 features: 0.94 (+/- 0.07)
* Accuracy KNN using best 40 features: 0.94 (+/- 0.07)
* **Accuracy KNN using best 60 features: 0.94 (+/- 0.04)**
* Accuracy KNN using best 70 features: 0.93 (+/- 0.06)

**Overall Results – Meth Data**

* Best accuracy showing SVM (linear kernel) here with all features. But it is not practical to use that much of features for prediction. And if we use that much of features there will be occur curse of dimensionality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of features** | **Cross Validation Score** | | |
| **SVM (Without transformation)** | **RF** | **KNN (Normalized with min max scaler)** |
| All (375385) | 0.96 (+/- 0.06) | 0.95 (+/- 0.04) | 0.71 (+/- 0.03) |
| 170 | 0.93 (+/- 0.08) | 0.92 (+/- 0.05) | 0.94 (+/- 0.05) |
| 130 | - | 0.94 (+/- 0.06) | - |
| 120 | - | 0.94 (+/- 0.05) | - |
| 110 | - | 0.94 (+/- 0.06) | - |
| 100 | 0.93 (+/- 0.06) | 0.93 (+/- 0.05) | 0.93 (+/- 0.06) |
| 90 | - | 0.93 (+/- 0.04) | - |
| 70 | - | - | 0.93 (+/- 0.06) |
| 60 | 0.92 (+/- 0.07) | 0.92 (+/- 0.05) | 0.94 (+/- 0.04) |
| 50 | 0.93 (+/- 0.06) | 0.93 (+/- 0.05) | 0.94 (+/- 0.06) |
| 40 | 0.93 (+/- 0.07) | 0.92 (+/- 0.05) | 0.94 (+/- 0.07) |
| 30 | 0.93 (+/- 0.06) | 0.92 (+/- 0.08) | 0.94 (+/- 0.07) |
| 20 | 0.91 (+/- 0.07) | 0.92 (+/- 0.07) | 0.92 (+/- 0.08) |

So, the second highest accuracy which we obtained around 94% using KNN can be choose as optimal model for Methylation data.

* Here below are the parameters we used for KNN
  + K=5
* Selected features (60):

|  |  |
| --- | --- |
| 1 | cg0586019 |
| 2 | cg1735603 |
| 3 | cg1220237 |
| 4 | cg0267012 |
| 5 | cg0626228 |
| 6 | cg0967733 |
| 7 | cg2423367 |
| 8 | cg2628069 |
| 9 | cg1260869 |
| 10 | cg2645004 |
| 11 | cg0515578 |
| 12 | cg1980516 |
| 13 | cg0182799 |
| 14 | cg1667222 |
| 15 | cg1134053 |
| 16 | cg2504225 |
| 17 | cg1907229 |
| 18 | cg0311544 |
| 19 | cg1290559 |
| 20 | cg1013100 |
| 21 | cg2090092 |
| 22 | cg0383058 |
| 23 | cg2349007 |
| 24 | cg0568714 |
| 25 | cg2158187 |
| 26 | cg2154463 |
| 27 | cg1628318 |
| 28 | cg0822300 |
| 29 | cg1624843 |
| 30 | cg0672706 |
| 31 | cg2344958 |
| 32 | cg0036475 |
| 33 | cg2348857 |
| 34 | cg1101173 |
| 35 | cg1824196 |
| 36 | cg0272106 |
| 37 | cg1516117 |
| 38 | cg0653174 |
| 39 | cg0027940 |
| 40 | cg1059302 |
| 41 | cg1455721 |
| 42 | cg2483245 |
| 43 | cg1535932 |
| 44 | cg1582750 |
| 45 | cg1078931 |
| 46 | cg2040221 |
| 47 | cg2697104 |
| 48 | cg0246466 |
| 49 | cg0603372 |
| 50 | cg2431558 |
| 51 | cg2385613 |
| 52 | cg0168417 |
| 53 | cg0788789 |
| 54 | cg1782519 |
| 55 | cg0197318 |
| 56 | cg1587137 |
| 57 | cg1655166 |
| 58 | cg2178678 |
| 59 | cg0020480 |
| 60 | cg1543165 |