Team Details: Group007

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Name (as appears in Canvas)** | **ID NO** |
| **01** | **ALKESH CHATURVEDI** | **2019AD04082** |
| **02** | **RAZA ABBAS** | **2019AD04095** |
| **03** | **HARSH KUMAR BHATIA K C BHATIA** | **2019AD04086** |
| **04** | **SHANUR RAHMAN** | **2019AD04065** |

Contents

[**1.** **Problem statement** 2](#_Toc34042563)

[**2.** **Exploratory Data Analysis** 2](#_Toc34042564)

[2(a) Descriptive statistics 2](#_Toc34042565)

[2(b) Correlation analysis and skewness determination 4](#_Toc34042566)

[2(c) Data Visualization 7](#_Toc34042567)

[**3.** **Preprocess the data** 9](#_Toc34042568)

[**4.** **Select training data, test data** 9](#_Toc34042569)

[**5.** **Train the Model** 10](#_Toc34042570)

[**6.** **Test the Model (Predictions and reporting)** 10](#_Toc34042571)

[**7.** **Evaluate the Model Performance** 10](#_Toc34042572)

[**8.** **Suggest ways of improving the model** 10](#_Toc34042573)

[**9.** **Final selection of model** 11](#_Toc34042574)

[**10.** **Challenges faced and how were they mitigated** 11](#_Toc34042575)

# **Problem statement**

The focus of this assignment will be **Problem2\_Data.csv** dataset. The problem is to predict if the person might have a coronary heart disease in the next ten years or not. To do this prediction, data (vitals collected in city hospitals) captured in aforementioned dataset will be used to construct a **classification model** and its efficiency will be evaluated. Finally, chosen model will be used to do the said prediction.

# **Exploratory Data Analysis**

## 2(a) Descriptive statistics

We first check the dimensions of the provided dataset. Example, the number of rows and columns for the given data in the provided dataset is checked. We have **(34281, 25)** returned on checking the shape. So, we have **34281** instances to work with along with **25** attributes including the class attribute.

Next we looked at the **datatype** of each of the attributes. We can see that all of the attributes are numeric (**int or float**).

|  |
| --- |
| import pandas as pd  filename = 'Problem2\_Data.csv'  df = pd.read\_csv(filename)  **print(df.dtypes)**  ID int64  IV int64  A1 int64  A2 float64  A3 int64  A4 int64  A5 int64  A6 int64  A7 int64  A8 int64  A9 int64  A10 int64  A11 int64  A12 int64  A13 int64  A14 int64  A15 float64  A16 float64  A17 int64  A18 int64  A19 int64  A20 int64  A21 float64  A22 int64  Target int64 |

**When we peek at the data given** in the provided dataset, we see that **ID column** represents patientID and it **should not have any correlation in predicting the Target attribute value**.

Likewise, it is noticed that **column A11 has '27' mentioned in every provided instance, so it also does not have any bearing with the Target attribute value determination.**

**Both ID and A11 columns can be removed from dataset as they do not have any impact on Target value derivation.**

**As we describe the data** (results shown below), we see that while **mean, median, 75 percentile** values of many columns are quite small, **maximum** value for these same columns is either too large or has large negative value as **minimum**. **This indicates that there is a high possibility of outliers in the data.**

|  |
| --- |
| print(df.describe())  ID IV A1 A2 A3 A4 \  count 3.428e+04 34281.000 34281.000 **32538.000** 34281.000 34281.000  mean 1.980e+06 **236.252 36.126 7.355 22.543 1.695**  std 6.385e+05 3326.575 427.707 6.165 359.486 36.196  min 1.059e+06 -2999.000 0.000 0.000 0.000 0.000  25% 1.464e+06 2.000 0.000 2.000 0.000 0.000  50% 1.842e+06 8.000 0.000 8.000 0.000 0.000  75% 2.254e+06 40.000 4.000 8.000 0.000 0.000  max 3.275e+06 **366924.000 50547.000** 52.000 **31750.000 2999.000**  A5 A6 A7 A8 A9 A10 \  count 34281.000 34281.000 34281.000 34281.000 34281.000 34281.000  mean **151.959 274.418 387.934 36.483 132.948 236.647**  std 2274.087 4065.441 5443.805 375.932 4151.795 4528.960  min 0.000 0.000 0.000 0.000 0.000 0.000  25% 0.000 0.000 0.000 0.000 0.000 0.000  50% 0.000 2.000 4.000 1.000 3.000 6.000  75% 18.000 33.000 48.000 6.000 19.000 37.000  max **260660.000 438020.000 533540.000 21071.000 742110.000 742750.000**  A11 A12 A13 A14 A15 A16 \  count 34281.0 34281.000 34281.000 34281.000 34281.000 34281.000  mean 27.0 **341.152** 0.002  **1.745**  -5.743 -5.369  std 0.0 5005.764 0.043 26.078 24.618 23.938  min 27.0 0.000 0.000 0.000 -99.000 -99.000  25% 27.0 0.000 0.000 0.000 0.590 0.610  50% 27.0 8.000 0.000 0.000 0.810 0.790  75% 27.0 54.000 0.000 0.000 0.960 0.940  max 27.0 **743215.000** 1.000  **1488.000**  1.000 1.000  A17 A18 A19 A20 A21 A22 \  count 34281.000 3.428e+04 34281.000 34281.000 34281.000 3.428e+04  mean 0.209 3.209e-04 0.132 0.962 17.050 2.042e-04  std 0.406 1.791e-02 0.338 0.190 0.029 1.429e-02  min 0.000 0.000e+00 0.000 0.000 17.000 0.000e+00  25% 0.000 0.000e+00 0.000 1.000 17.025 0.000e+00  50% 0.000 0.000e+00 0.000 1.000 17.050 0.000e+00  75% 0.000 0.000e+00 0.000 1.000 17.075 0.000e+00  max 1.000 1.000e+00 1.000 1.000 17.100 1.000e+00  Target  count 34281.000  mean 0.329  std 0.470  min 0.000  25% 0.000  50% 0.000  75% 1.000  max 1.000 |

**We will determine the outliers using Box plot visualization as well and will investigate these outliers further and will take corrective measures.**

**Null value replacement**

**A2 column has Null values in many instances which are replaced by its mean value. We have chosen mean value for replacement because there was not much skewness for A2 column.**

**Similarly, A15 & A16 columns have Null values shown with -99 value in the given data. They are replaced by its median value as data has negative skewness.**

Now as we look at the breakdown of class values, we get the below information:-

|  |
| --- |
| class\_counts = df.groupby('Target').size()  print(class\_counts)  Target  0 22988  1 11293 |

This shows that the **classes are highly imbalanced between 0 and 1**. This is expected as this being medical data which depicts most of the patients as healthy (Target value of 0: have no coronary heart disease symptoms).

**We will take special consideration while choosing training and test data for the model to have balanced classes in the training data and later on in the test data.**

## 2(b) Correlation analysis and skewness determination

We will find correlation and skewness in provided data as show below

**Correlation**

|  |
| --- |
| correlations = df.corr(method='pearson')  print(correlations)  ID IV A1 A2 A3 A4 \  ID 1.000 2.381e-03 8.931e-03 2.667e-02 1.243e-02 -1.004e-02  IV 0.002 1.000e+00 1.812e-01 2.714e-02 1.683e-01 -9.892e-03  A1 0.009 1.812e-01 1.000e+00 -6.628e-03 3.939e-01 4.505e-02  A2 0.027 2.714e-02 -6.628e-03 1.000e+00 -6.445e-03 9.243e-03  A3 0.012 1.683e-01 3.939e-01 -6.445e-03 1.000e+00 1.555e-02  A4 -0.010 -9.892e-03 4.505e-02 9.243e-03 1.555e-02 1.000e+00  A5 0.003 1.306e-01 7.581e-01 -6.401e-03 2.964e-01 7.851e-02  A6 0.004 1.961e-01 7.721e-01 -6.888e-03 3.539e-01 5.992e-02  A7 0.004 2.388e-01 7.589e-01 -7.617e-03 3.818e-01 5.924e-02  A8 0.007 2.273e-01 5.807e-01 -1.534e-02 5.292e-01 7.536e-02  A9 0.011 6.443e-02 1.704e-01 9.905e-04 1.490e-01 2.346e-02  A10 0.011 1.370e-01 2.970e-01 -1.348e-03 2.665e-01 3.683e-02  A11 NaN NaN NaN NaN NaN NaN  A12 0.011 1.642e-01 3.736e-01 -3.146e-03 3.539e-01 4.361e-02  A13 -0.004 -2.919e-03 9.923e-03 -4.918e-03 8.601e-03 1.622e-02  A14 -0.011 4.802e-03 7.353e-02 -1.100e-02 7.372e-02 3.546e-02  A15 -0.021 -2.746e-02 -8.643e-03 -6.683e-02 5.612e-03 8.419e-03  A16 -0.022 -2.672e-02 -7.344e-03 -7.901e-02 6.327e-03 8.462e-03  A17 -0.009 -1.514e-02 -3.410e-02 6.285e-02 -2.841e-02 7.045e-03  A18 0.002 -1.063e-03 8.610e-05 -9.466e-03 -2.038e-04 -7.042e-04  A19 -0.021 -5.293e-03 1.415e-02 -1.309e-03 -3.313e-03 -1.508e-03  A20 -0.002 -1.628e-02 -1.393e-02 -2.054e-01 1.448e-03 3.634e-03  A21 0.005 -5.399e-03 4.666e-04 -2.733e-03 -2.475e-03 1.205e-03  A22 0.005 -8.572e-04 4.335e-03 -3.403e-05 -8.962e-04 -6.694e-04  Target -0.250 -4.526e-02 -1.951e-02 -1.194e-01 -3.549e-02 5.492e-02  A5 A6 A7 A8 A9 A10 A11 \  ID 2.860e-03 4.370e-03 3.777e-03 7.470e-03 1.114e-02 1.103e-02 NaN  IV 1.306e-01 1.961e-01 2.388e-01 2.273e-01 6.443e-02 1.370e-01 NaN  A1 7.581e-01 7.721e-01 7.589e-01 5.807e-01 1.704e-01 2.970e-01 NaN  A2 -6.401e-03 -6.888e-03 -7.617e-03 -1.534e-02 9.905e-04 -1.348e-03 NaN  A3 2.964e-01 3.539e-01 3.818e-01 5.292e-01 1.490e-01 2.665e-01 NaN  A4 7.851e-02 5.992e-02 5.924e-02 7.536e-02 2.346e-02 3.683e-02 NaN  A5 1.000e+00 9.702e-01 9.475e-01 2.928e-01 8.709e-02 1.562e-01 NaN  A6 **9.702e-01** 1.000e+00 9.913e-01 3.419e-01 1.010e-01 1.834e-01 NaN  A7 **9.475e-01** **9.913e-01** 1.000e+00 3.763e-01 1.109e-01 2.016e-01 NaN  A8 2.928e-01 3.419e-01 3.763e-01 1.000e+00 2.458e-01 4.180e-01 NaN  A9 8.709e-02 1.010e-01 1.109e-01 2.458e-01 1.000e+00 **9.743e-01** NaN  A10 1.562e-01 1.834e-01 2.016e-01 4.180e-01 9.743e-01 1.000e+00 NaN  A11 NaN NaN NaN NaN NaN NaN NaN  A12 2.085e-01 2.475e-01 2.722e-01 5.232e-01 **9.239e-01 9.836e-01** NaN  A13 4.988e-02 3.628e-02 3.485e-02 1.379e-02 3.316e-03 5.858e-03 NaN  A14 9.157e-02 7.948e-02 8.242e-02 1.002e-01 2.825e-02 4.903e-02 NaN  A15 1.285e-02 1.150e-02 1.031e-02 -1.114e-02 -1.629e-03 -2.257e-03 NaN  A16 1.304e-02 1.150e-02 1.046e-02 -7.753e-03 -9.793e-04 -1.250e-03 NaN  A17 -2.035e-02 -2.275e-02 -2.528e-02 -4.051e-02 -1.379e-02 -2.259e-02 NaN  A18 1.228e-04 1.051e-04 1.558e-04 4.622e-04 1.693e-04 3.578e-04 NaN  A19 8.585e-03 6.578e-03 5.785e-03 5.955e-03 -2.324e-04 4.975e-04 NaN  A20 1.676e-03 3.142e-03 3.666e-03 -2.206e-02 -4.913e-03 -9.291e-03 NaN  A21 3.505e-03 8.877e-04 5.638e-04 -7.753e-04 -7.692e-04 -1.553e-03 NaN  A22 -9.550e-04 -9.647e-04 -1.018e-03 -1.197e-03 -1.572e-04 -2.896e-04 NaN  Target 1.651e-03 -5.127e-03 -7.982e-03 -1.327e-02 -9.048e-03 -1.507e-02 NaN  A12 A13 A14 A15 A16 A17 A18 \  ID 1.058e-02 -3.516e-03 -1.137e-02 -0.021 -2.174e-02 -0.009 2.331e-03  IV 1.642e-01 -2.919e-03 4.802e-03 -0.027 -2.672e-02 -0.015 -1.063e-03  A1 3.736e-01 9.923e-03 7.353e-02 -0.009 -7.344e-03 -0.034 8.610e-05  A2 -3.146e-03 -4.918e-03 -1.100e-02 -0.067 -7.901e-02 0.063 -9.466e-03  A3 3.539e-01 8.601e-03 7.372e-02 0.006 6.327e-03 -0.028 -2.038e-04  A4 4.361e-02 1.622e-02 3.546e-02 0.008 8.462e-03 0.007 -7.042e-04  A5 2.085e-01 4.988e-02 9.157e-02 0.013 1.304e-02 -0.020 1.228e-04  A6 2.475e-01 3.628e-02 7.948e-02 0.011 1.150e-02 -0.023 1.051e-04  A7 2.722e-01 3.485e-02 8.242e-02 0.010 1.046e-02 -0.025 1.558e-04  A8 5.232e-01 1.379e-02 1.002e-01 -0.011 -7.753e-03 -0.041 4.622e-04  A9 9.239e-01 3.316e-03 2.825e-02 -0.002 -9.793e-04 -0.014 1.693e-04  A10 9.836e-01 5.858e-03 4.903e-02 -0.002 -1.250e-03 -0.023 3.578e-04  A11 NaN NaN NaN NaN NaN NaN NaN  A12 1.000e+00 5.951e-03 6.574e-02 -0.001 1.879e-04 -0.030 3.766e-04  A13 5.951e-03 1.000e+00 2.873e-02 0.008 8.051e-03 -0.002 3.726e-02  A14 6.574e-02 2.873e-02 1.000e+00 0.005 5.177e-03 -0.021 1.549e-03  A15 -1.323e-03 8.439e-03 5.112e-03 1.000 9.687e-01 -0.191 4.373e-03  A16 1.879e-04 8.051e-03 5.177e-03 **0.969** 1.000e+00 -0.195 4.281e-03  A17 -3.017e-02 -1.920e-03 -2.131e-02 -0.191 -1.947e-01 1.000 2.825e-03  A18 3.766e-04 3.726e-02 1.549e-03 0.004 4.281e-03 0.003 1.000e+00  A19 1.090e-03 2.755e-02 1.125e-02 -0.025 -2.717e-02 0.064 -2.172e-03  A20 -1.337e-02 8.485e-03 -9.571e-03 0.271 2.681e-01 -0.113 3.543e-03  A21 -2.130e-03 -7.044e-03 5.345e-03 -0.003 -4.985e-03 -0.001 -6.401e-03  A22 -4.960e-04 -6.132e-04 -9.562e-04 -0.038 -3.889e-02 -0.007 -2.560e-04  Target -1.887e-02 4.383e-02 4.793e-02 0.075 7.559e-02 -0.067 1.517e-02  A19 A20 A21 A22 Target  ID -2.140e-02 -0.002 4.929e-03 4.872e-03 -0.250  IV -5.293e-03 -0.016 -5.399e-03 -8.572e-04 -0.045  A1 1.415e-02 -0.014 4.666e-04 4.335e-03 -0.020  A2 -1.309e-03 -0.205 -2.733e-03 -3.403e-05 -0.119  A3 -3.313e-03 0.001 -2.475e-03 -8.962e-04 -0.035  A4 -1.508e-03 0.004 1.205e-03 -6.694e-04 0.055  A5 8.585e-03 0.002 3.505e-03 -9.550e-04 0.002  A6 6.578e-03 0.003 8.877e-04 -9.647e-04 -0.005  A7 5.785e-03 0.004 5.638e-04 -1.018e-03 -0.008  A8 5.955e-03 -0.022 -7.753e-04 -1.197e-03 -0.013  A9 -2.324e-04 -0.005 -7.692e-04 -1.572e-04 -0.009  A10 4.975e-04 -0.009 -1.553e-03 -2.896e-04 -0.015  A11 NaN NaN NaN NaN NaN  A12 1.090e-03 -0.013 -2.130e-03 -4.960e-04 -0.019  A13 2.755e-02 0.008 -7.044e-03 -6.132e-04 0.044  A14 1.125e-02 -0.010 5.345e-03 -9.562e-04 0.048  A15 -2.524e-02 0.271 -3.064e-03 -3.759e-02 0.075  A16 -2.717e-02 0.268 -4.985e-03 -3.889e-02 0.076  A17 6.432e-02 -0.113 -1.169e-03 -7.338e-03 -0.067  A18 -2.172e-03 0.004 -6.401e-03 -2.560e-04 0.015  A19 1.000e+00 -0.043 1.246e-03 6.494e-03 0.049  A20 -4.294e-02 1.000 -2.333e-03 -1.863e-02 -0.015  A21 1.246e-03 -0.002 1.000e+00 9.254e-03 0.004  A22 6.494e-03 -0.019 9.254e-03 1.000e+00 -0.010  Target 4.934e-02 -0.015 3.502e-03 -1.002e-02 1.000 |

Above correlation information can be inferred as below:-

A5 & A7 columns are correlated as the correlation value is close to 1 and shown above in **red**.

Similarly, below columns are also highly correlated.

A5 & A6 columns are correlated.

A6 & A7 columns are correlated.

A9 & A10 columns are correlated.

A9 & A12 columns are correlated.

A10 & A12 columns are correlated.

A15 & A16 columns are correlated.

Also, same can be seen in the heat map generated for the correlation.

**Correlation** data visualization **chart** is shown below.

|  |
| --- |
|  |

**Based on the correlation results mentioned above, we are removing columns A6, A7, A10, A12 and A16.**

**Skewness**

|  |
| --- |
| print(df.skew().astype(int))  ID 0  IV 69  A1 64  A2 4  A3 56  A4 53  A5 66  A6 60  A7 53  A8 34  A9 167  A10 132  A11 0  A12 102  A13 23  A14 31  A15 -3  A16 -3  A17 1  A18 55  A19 2  A20 -4  A21 0  A22 69  Target 0 |

Skewness data shown above depicts a large positive or somewhat negative skew. We will check the skewness again after treating the outliers.

## 2(c) Data Visualization

We visualized data using **box and whisker plots** to get an idea of the spread of the values.

|  |
| --- |
|  |

**We clearly see outliers in the box plot above. After removing those outliers which are beyond the 6 sigma limits, below is the visualization of data and it looks much better.**

|  |
| --- |
|  |

**After getting encouraging results in the above pasted Box plot, we have decided to go ahead with the approach of removing outlier data from the given dataset. Also skewness has gone after outliers removal.**

Here, in the below pasted 3D plots, we can see that when we take any set of 3 different attributes from the provided dataset, in most of the cases, we can clearly distinguish the data in two Target classes (0 or 1). But, at the same time, in some cases, data pertaining to both the class labels is so densly intermingled that it is quite difficult to classify any new data point. This may lead to misclassification if we pick a specific classification algorithm to solve this problem. Hence, we are required to pick from a list of different classifiers, as done finally to solve this problem. Finally, classifier with best accuracy will be picked.

|  |
| --- |
|  |

# **Preprocess the data**

We are normalizing the given data present in different columns to bring it at the comparable scale.

|  |
| --- |
| from sklearn.preprocessing import Normalizer  scaler = Normalizer().fit(X\_train)  standarized\_x = scaler.transform(X\_train)  standarized\_x\_test = scaler.transform(X\_test) |

Observations taken care before this step are:-

* Remove outlier values
* Replace NaN/-99 values from data with mean or median values
* Remove additional attribute columns from given data as they are correlated (A6, A7, A10, A12 and A16)
* Remove columns ID and A11 ( as it has static value of '27') from the processing

# **Select training data, test data**

We are using 80% of the data for modelling and holding back 20% for test. As class count is imbalanced, we are ensuring that 80% of chosen data for modelling has enough training data for either of the classes. Likewise, we are taking care of test data too.

|  |
| --- |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, **test\_size=0.2**, random\_state=42) |

# **Train the Model**

Initially, we have chosen RandomForestClassifier and trained it with 80% of the data held for training purposes.

# **Test the Model (Predictions and reporting)**

**RandomForestClassifier** model is tested with 20% of held back data for test and model **gave an accuracy of 90.01%**

# **Evaluate the Model Performance**

As we did not know which algorithms will do best on the given dataset, we tried different algorithms used for classification problems viz, Gaussian NB, SVM, KNN & Logistic regression. **We used 10-fold cross validation**. The dataset is not too small and this is a good standard test harness conﬁguration. **We evaluated algorithms using the accuracy metric and found below data.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **10 -fold cross validation accuracy** | **Value** | **Precision** | **Recall** | **F1-score** |
| **RandomForest** | 90.05% | 88.48% | 0 | 0.93 | 0.92 | 0.93 |
| 1 | 0.84 | 0.85 | 0.84 |
| **GaussianNB** | 59.59% | 64.43% | 0 | 0.97 | 0.43 | 0.59 |
| 1 | 0.44 | 0.97 | 0.6 |
| **SVM** | 85.72% | 86.60% | 0 | 0.89 | 0.91 | 0.9 |
| 1 | 0.79 | 0.75 | 0.77 |
| **KNN** | 87.96% | 88.04% | 0 | 0.93 | 0.89 | 0.91 |
| 1 | 0.78 | 0.86 | 0.82 |
| **Logistic Regression** | 86.20% | 83.91% | 0 | 0.89 | 0.92 | 0.9 |
| 1 | 0.8 | 0.74 | 0.77 |

# **Suggest ways of improving the model**

Algorithms tuning is one of the ways to improve model accuracy. **We have used KNN tuning and SVM tuning for that.** Another way is to use ensemble methods like RF (Random Forest) which we already used and got maximum accuracy of 90.05% with it. Apart from this, all points listed in step 3 above are considered for improving the model accuracy, that is, remove outliers, replace NaN/-99 values with mean/median values, remove correlated columns and remove columns with no significance in changing the Target attribute value like ID, A11 (having constant value of '27' in it).

|  |
| --- |
| **KNN tuning results**  In KNN tuning, we have used elbow method to find the most appropriate K value which gives the least error rate. We choose a K value of 3 using this method. When we run KNN algorithm with K value of 3, we got an accuracy of 88.04% which was better than earlier obtained accuracy of 87.96%.  However this accuracy of 88.04% obtained at K value of 3 is still much less than accuracy of 90.05% obtained using RF classifier.    **SVM tuning results**  **GridSearch method** is used to determine the most appropriate values for C and gamma parameters (regularization parameter) in SVM classifier. We got a value of C=10 and gamma = 0.001 and with these values, SVM gave an accuracy of 86.60% which is better than previously obtained accuracy of 85.72% (without C and gamma parameters).  Again, this accuracy of 86.60% is much less than accuracy of 90.05% obtained using RF classifier. |

# **Final selection of model**

Based on accuracy of different algorithms, we have chosen RandomForest classifier. Below is the accuracy, f-score, precision and recall of the chosen model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Value** | **Precision** | **Recall** | **F1-score** |
| **RandomForest** | 90.05% | 0 | 0.93 | 0.92 | 0.93 |
| 1 | 0.84 | 0.85 | 0.84 |

**Any new data received will be first pre-processed. Finalized data will be passed to this model to predict the Target attribute value.**

# **Challenges faced and how were they mitigated**

Following are the challenges encountered during this assignment:-

1. We noticed many data quality issues in the provided dataset. For ex. presence of many outliers. Outliers were finally removed from the data.
2. Provided dataset had many NaN and -99 values. These values were replaced by their mean/median values.
3. Column A11 has static data value '27' in it and this column is altogether removed. Also ID column is removed considering it has no relation to change Target column value.
4. As provided dataset has lot of correlated columns, it was a big problem for the speed at which model is trained. Also, having redundant data is of no use. So we decided to remove correlated columns as well from the data.
5. We did not know the right classification algorithm beforehand that can work best with the provided data. To mitigate this problem, we trained and tested different algorithms and analyzed their individual performances. Finally we decided to go with RandomForest classifier as a solution for this problem.