Report on
Design and Application of a Machine Learning System for a Practical Problem

WORD COUNT: 1435

Lokesh Palaniraj | Reg - 2211543 | CE802 – MACHINE LEARNING

University of Essex

Introduction

The report aimed on the exploration of the performance of different machine learning models in the prediction of patients who might be at higher risk of developing diabetes and as well as the blood pressure level. The models in part 2 and part 3 were performed and evaluated with visualization to assess the performance of the solution of various different models. This report will provide a summary of all the models and their analysis on the comparison of their efficiency. We have used many ML models like Decision tree, Linear regression, Random Forest, kNN, SVM, Gradient Boosting for both problems for detecting the severity of diabetes (classification) and the blood glucose level (regression).

Life cycle of a study:

1. Data Collection & Preprocessing
2. Model Selection
3. Fitting the model with training data
4. Evaluating the model's performance
5. Adjusting the model
6. Prediction phase

Architecture Design & Data Preprocessing

We have used scikit learn package to import all the possible machine learning models and to evaluate their accuracy. Here is the list of packages we used for the comparative studies: numpy, pandas, matplotlib, sklearn, missingno, seaborn. We dealt with the missing values in three different approaches such as the baseline approach (removing the feature with blank values), BayesianRidge and the kNN imputation method. As we had 50% of the data missing in the feature, we couldn't go with the mean/median imputation, as those would be inefficient.

Part II - Comparative Study

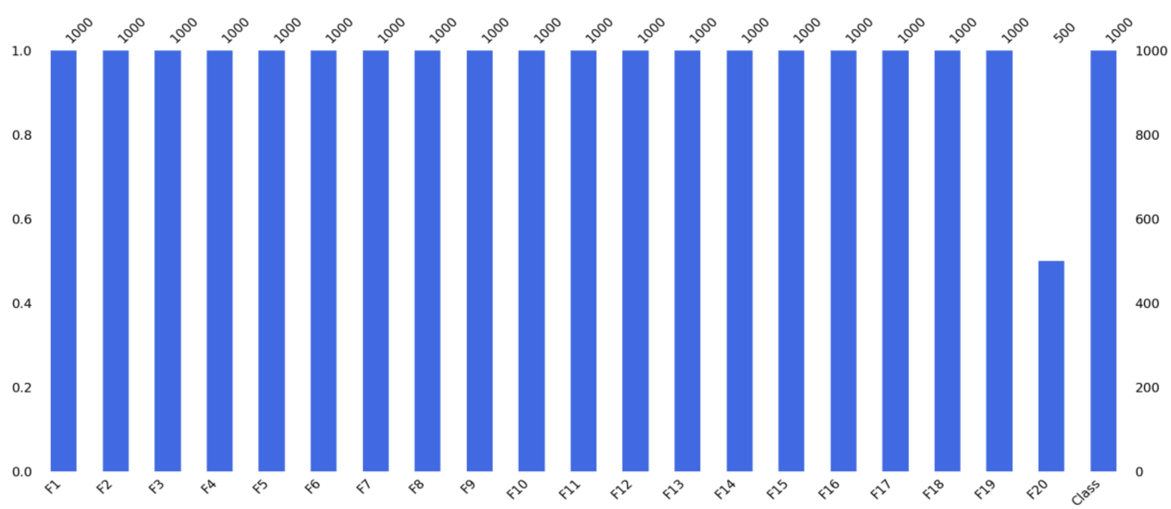
The main objective of this study is to predict whether the patients are at high risk of developing diabetes or not. We have a dataset full of patient details with the required features, along with those who were diagnosed with diabetes and who were not. So, basically the task is a **binary classification** task.

Initially, reviewing the description of the dataset with the count, mean, standard deviation and the quantiles to get an overall idea on it.

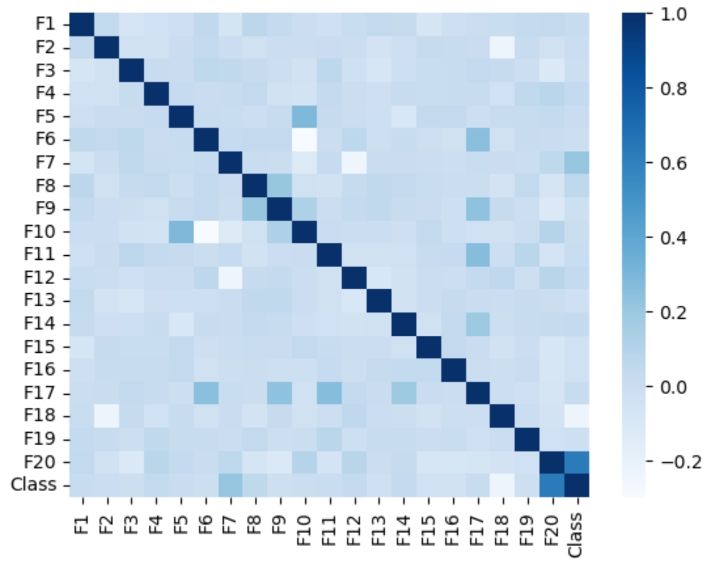
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	1.397000	-4.695798	-1.653968	11.573618	6.108717	5515.453754	11128.128917	-5.058608	-2143.218490	6594.407898
std	0.500201	2.699756	0.765717	2.708288	1.739357	1534.747192	1587.054314	0.901204	702.890861	1494.941683
min	0.890000	-14.976000	-7.209000	8.706600	4.264440	-4924.080000	1686.600000	-8.675000	-5873.260000	-3367.200000
25%	0.890000	-6.144750	-1.843675	9.559425	4.806650	5108.820000	10513.050000	-5.427000	-2424.337500	6432.700000
50%	1.890000	-3.824400	-1.348400	10.721100	5.598100	5482.585500	10770.135000	-4.777250	-2281.570000	6971.700000
75%	1.890000	-2.590725	-1.178988	12.762750	6.879000	5917.620000	11235.600000	-4.381825	-2049.885000	7225.106500
max	1.890000	-1.774140	-1.130002	22.176000	13.290000	17287.920000	27822.600000	-4.122990	2671.740000	14678.800000

	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	500.000000
mean	-3.740958	2.520013	0.508000	-120989.462060	-2.930605	0.516000	-2592.387215	-16777.828266	90.724599	31.280200
std	0.902777	0.853456	0.500186	5064.689413	0.606601	0.499994	493.077434	2074.355688	20.571395	2.311671
min	-7.238000	1.621290	0.000000	-279151.140000	-5.964000	0.000000	-5874.120000	-32323.100000	70.424656	24.560000
25%	-4.153500	1.889000	0.000000	-120901.470000	-3.265750	0.000000	-2766.202500	-17004.850000	76.804000	29.680000
50%	-3.462250	2.248750	1.000000	-120843.480000	-2.792350	1.000000	-2700.060000	-16364.700000	84.584000	31.240000
75%	-3.047450	2.885000	1.000000	-120787.035000	-2.446800	1.000000	-2565.445000	-15977.540000	96.325000	32.830000
max	-2.752120	6.073000	1.000000	-109690.140000	-2.170056	1.000000	587.880000	-7663.100000	233.980000	38.040000

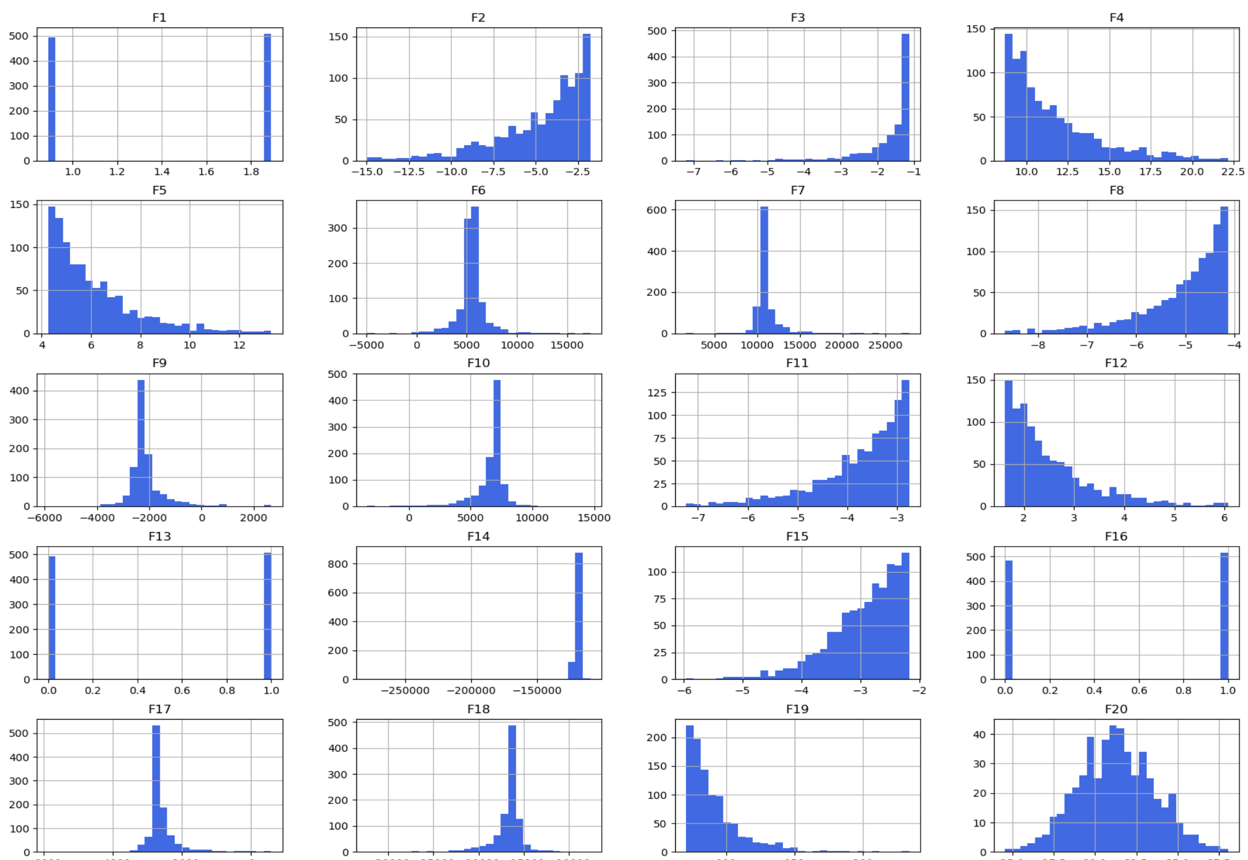
We can see that half of the data from 'F20' column was missing, and it should be managed in order to anticipate the target efficiently & to attain at most performance. There are many imputation methods to fill the blank values and those were covered in the report that follows.

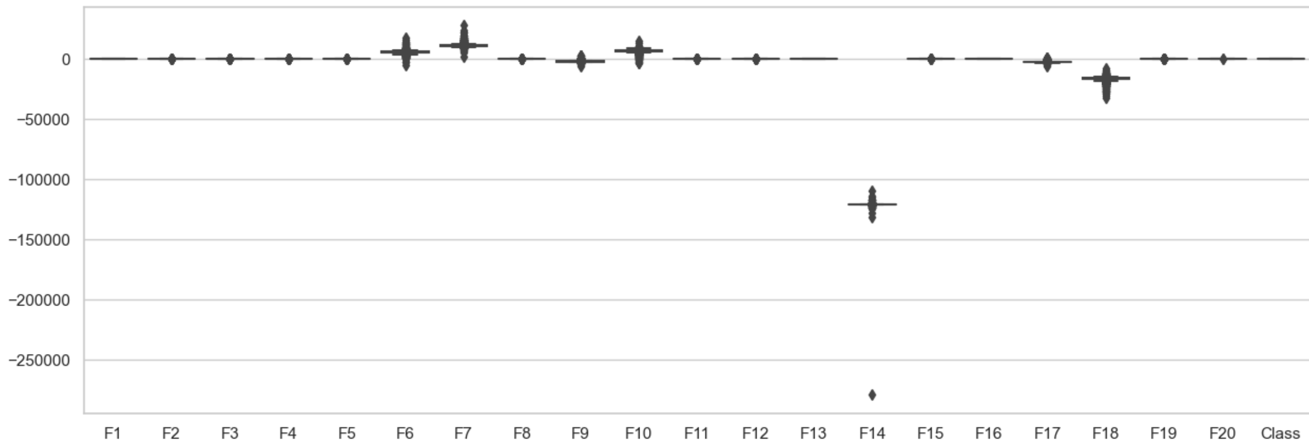


Now, looking at the correlation of features from the dataset using the seaborn package, we can observe that the feature F20 possesses at most correlation with the target variable 'class' from the dataset.



Looking at the distribution of data in each feature, we can examine the pattern (skewness/symmetric) of each column and how they are divided and presenting a box plot to identify the outliers.





So far, we have examined the data, now we are going to treat column F20 with the blank values by ***three*** different methods i.e.,

1. baseline approach (entirely removing the feature),
2. BayesianRidge Iterative Imputer to fill the empty values,
3. kNN imputer to fill the empty values.

We will be using ***four*** different ***machine learning models*** to predict the target variable i.e.,

1. Decision Tree Classifier
2. k Nearest Neighbor Classifier
3. Support Vector Machine Classifier
4. Random Forest Classifier

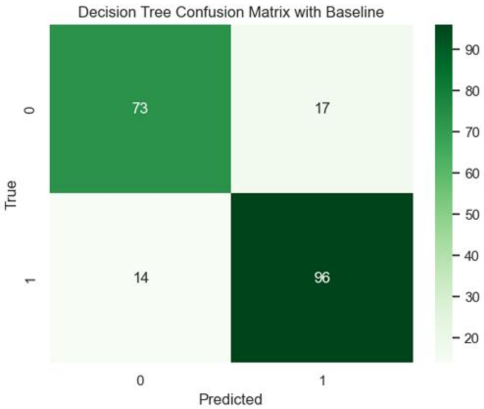
1. Baseline Approach:

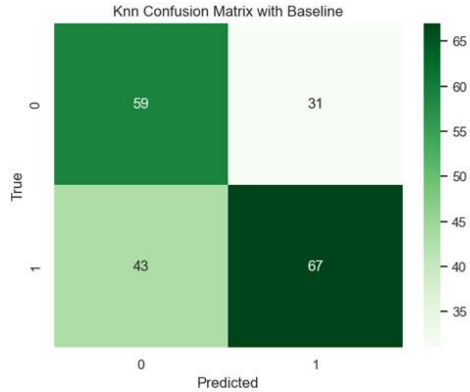
Here in this approach, we'll be removing the entire column with blank values ('F20') and scaling the data with the *StandardScalar* to standardize all the features in the dataset using the formula below.

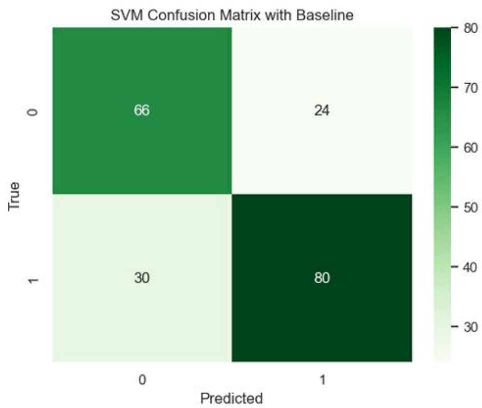
$$z = (x - \mu)/\sigma$$

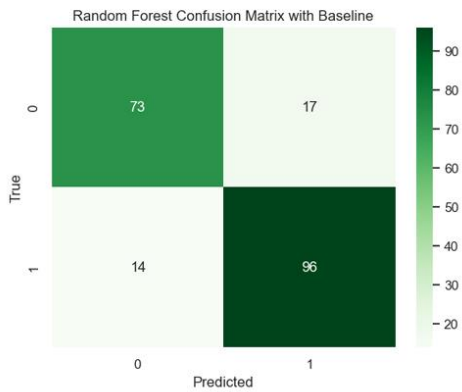
The data has been split into 80% for the training and 20% for the testing using *train_test_split()* and converted the categorical feature 'Class' into numerical using *LabelEncoder()*. Then the model has been fitted with the training data and has been predicted using the test data and to check the *performance* of the model we'll be using the **confusion matrix**, which has been presented below.

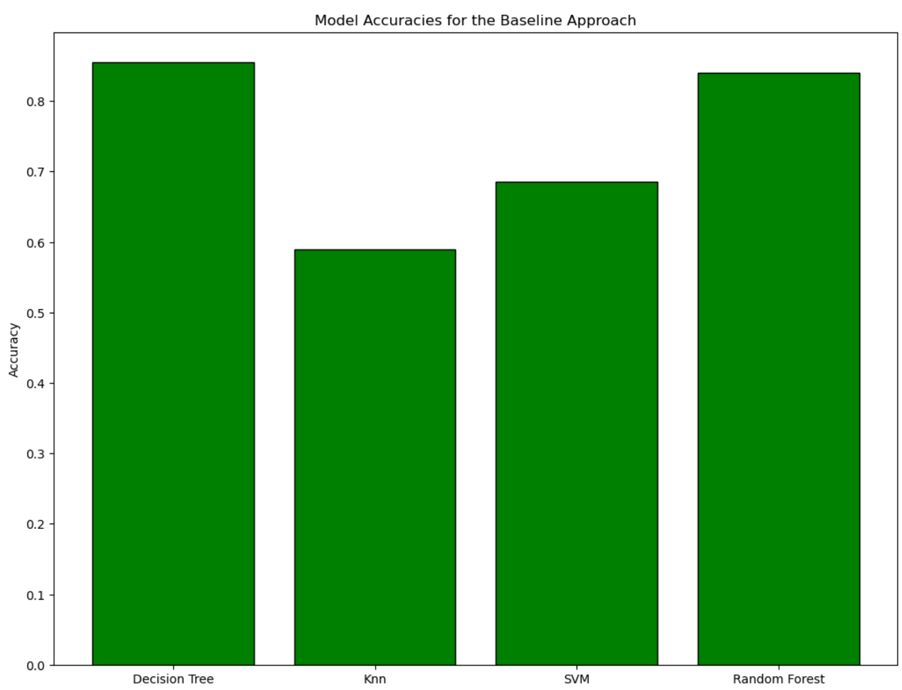
BASELINE APPROACH







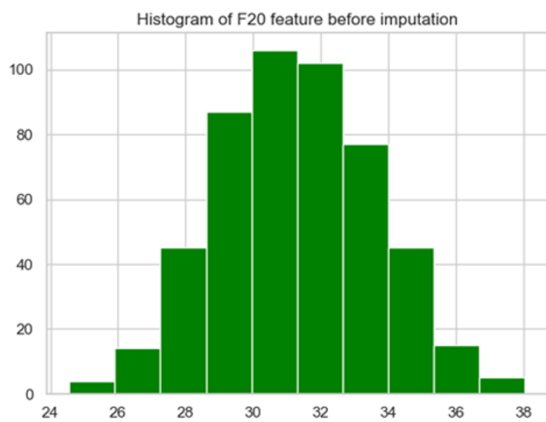


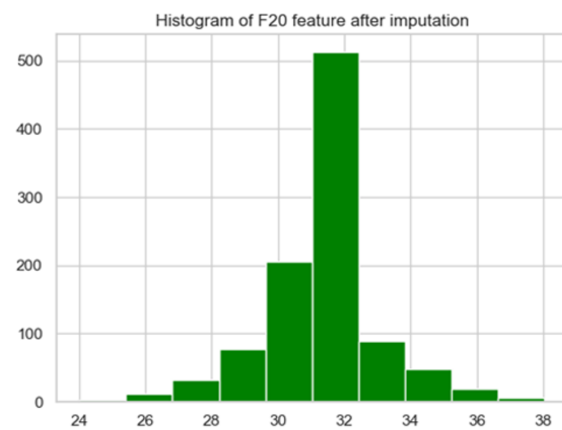


From the histogram above, we can tell that decision tree attained the most accuracy of 86%.

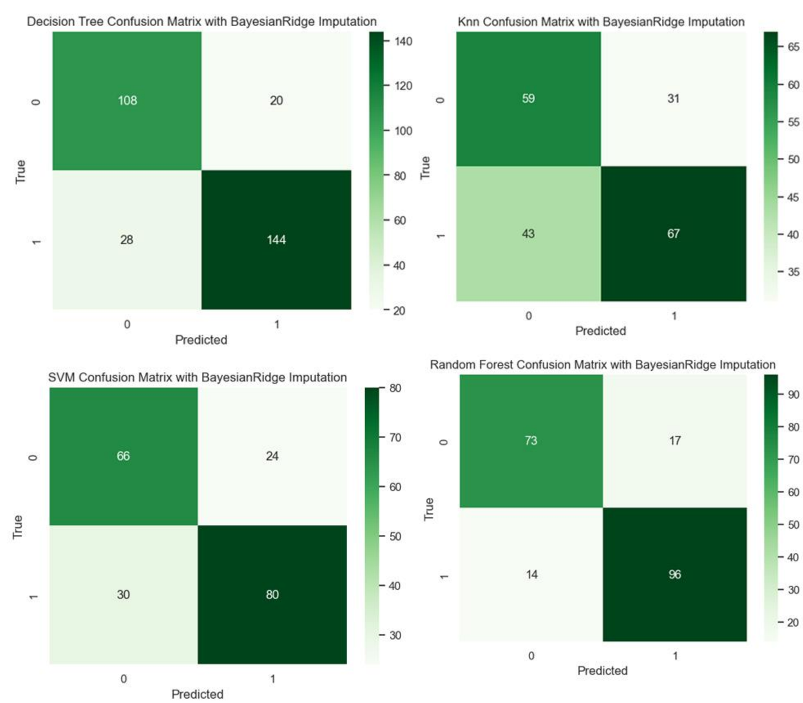
2. BayesianRidge Imputation:

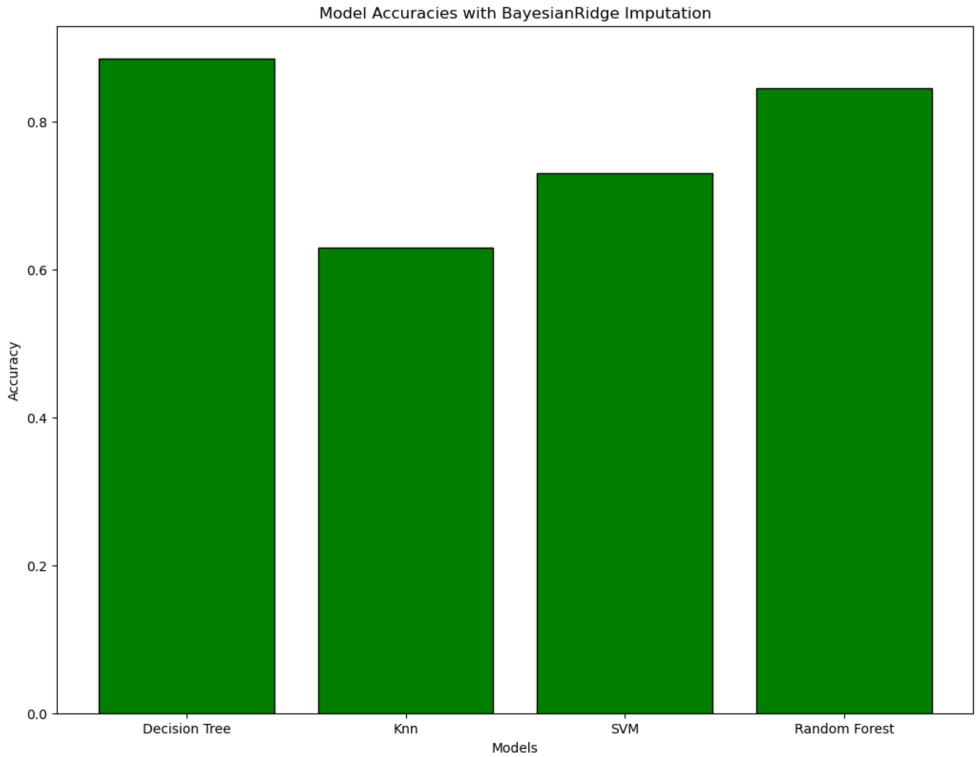
For the next step, we'll be using an iterative imputer called BayesianRidge to fill the empty values in the column 'F20'. It works by designing a statistical model which in turn predicts the empty cells and fills it using a priori assumption and the posteriori observation using mean/median of the distribution, it continues until the blank values are filled and follows by the scaling and splitting.





BayesianRidge Imputation

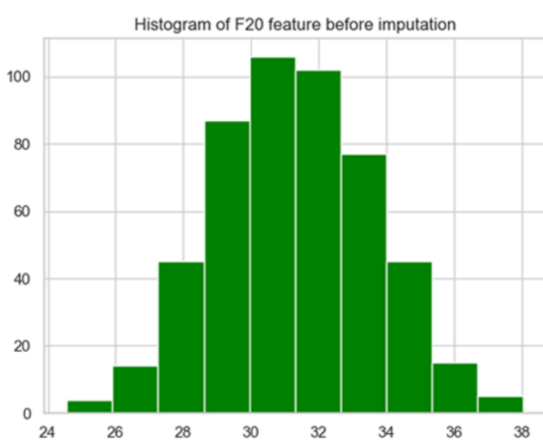


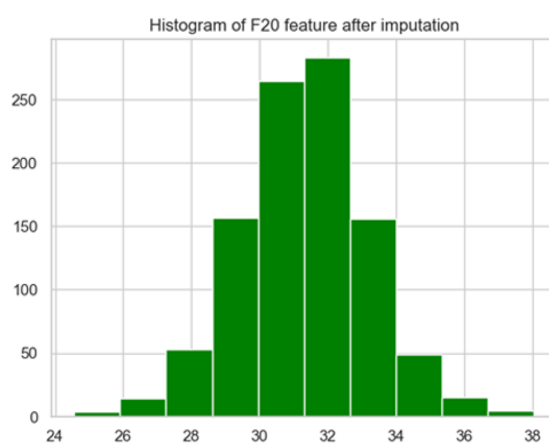


This imputation did improve the efficiency of the model and increased the accuracy of decision tree to 88%, which has been assessed using the confusion matrix.

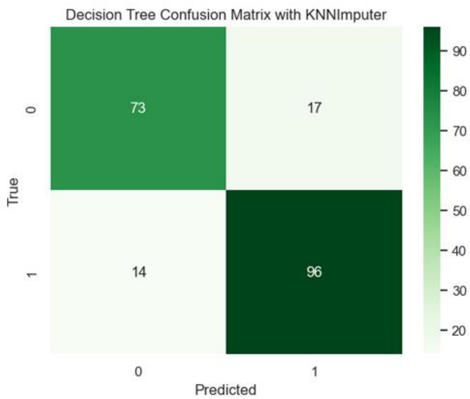
3. kNN imputation:

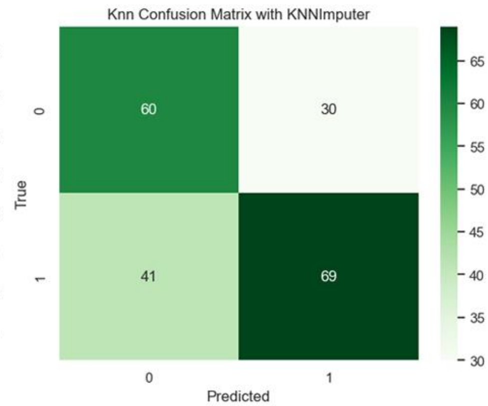
In this imputation method, we'll be managing the NULL values using kNN imputer, which basically identifies the neighbors close to the missing values which are known as the k-nearest neighbors and takes the average of these values to replace it with the missing values, this is repeated until all the vacancies are filled.

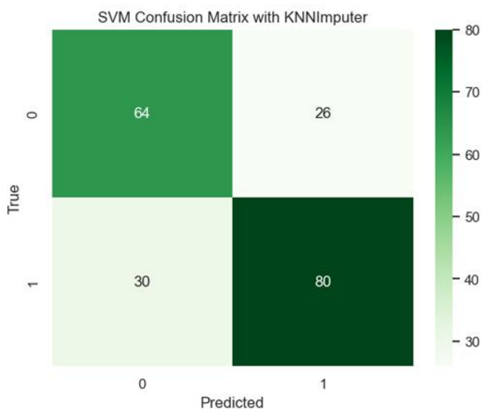


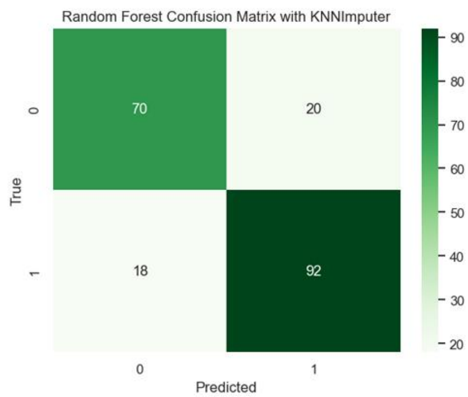


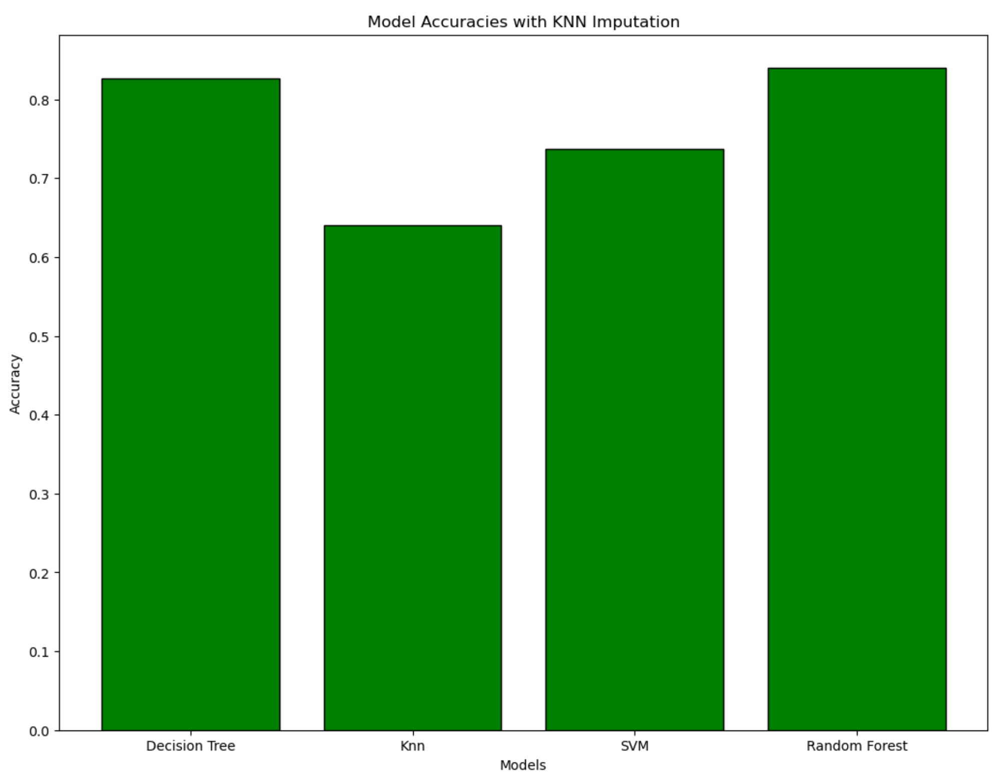
kNN Imputation



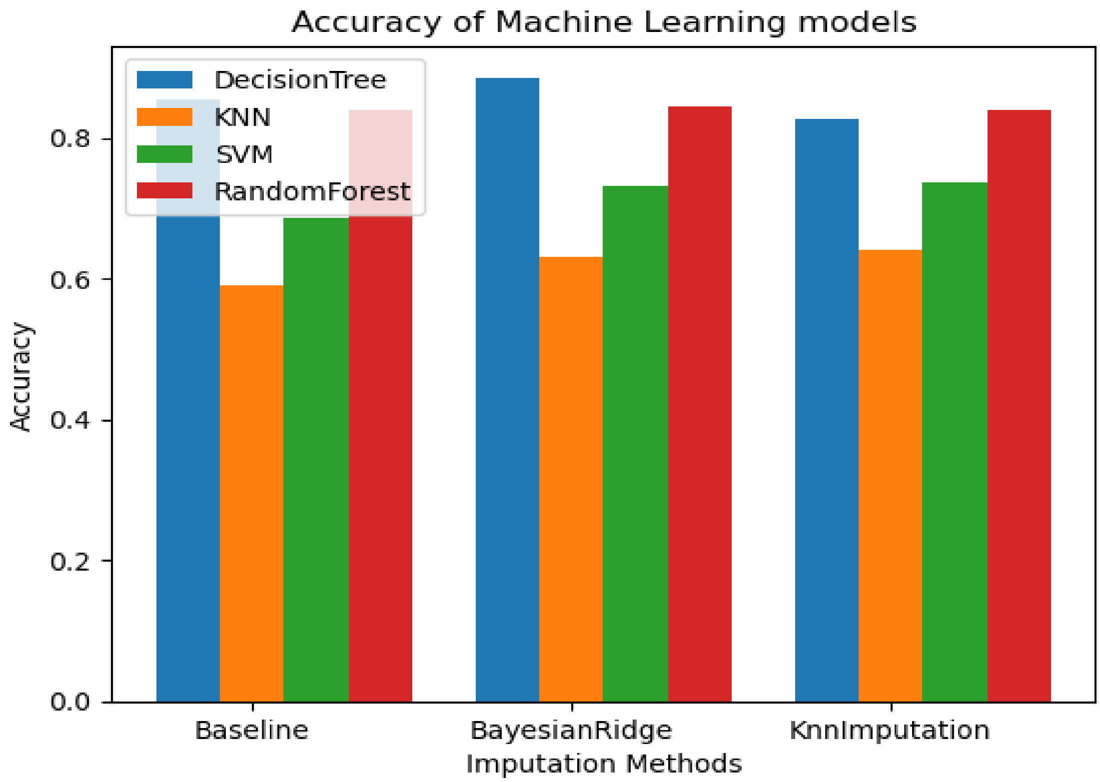








By using the kNN imputer we can be able to achieve 83% accuracy in the Decision Tree model and 81% in Random Forest respectively. Overall, when looking the performance of the models in all three ways of filling the missing values, the Decision Tree peaked with 88% accuracy when dealt with BayesianRidge imputation and the remaining are visualized below for better interpretation.



For the next section of the comparative study, we used the trained decision tree model (which was the best performed) from the previous section and predicted the patients with higher risk of developing diabetes and the results are as follows:

Number of patients with higher risk of diabetes: 470

Number of patients with lower risk of diabetes: 530

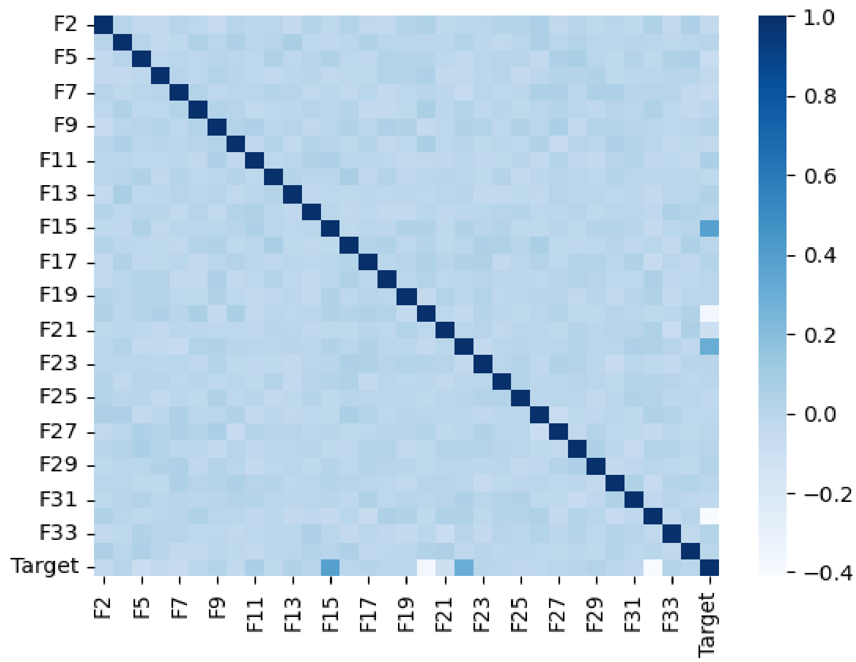
Part III - Additional Comparative Study

In this study, we'll be predicting whether the patient is at higher risk of developing diabetes by calculating the blood glucose level and checking whether it exceeds the threshold. So basically, we must predict the blood glucose level of each patient with the corresponding features provided, which is a **regression problem**.

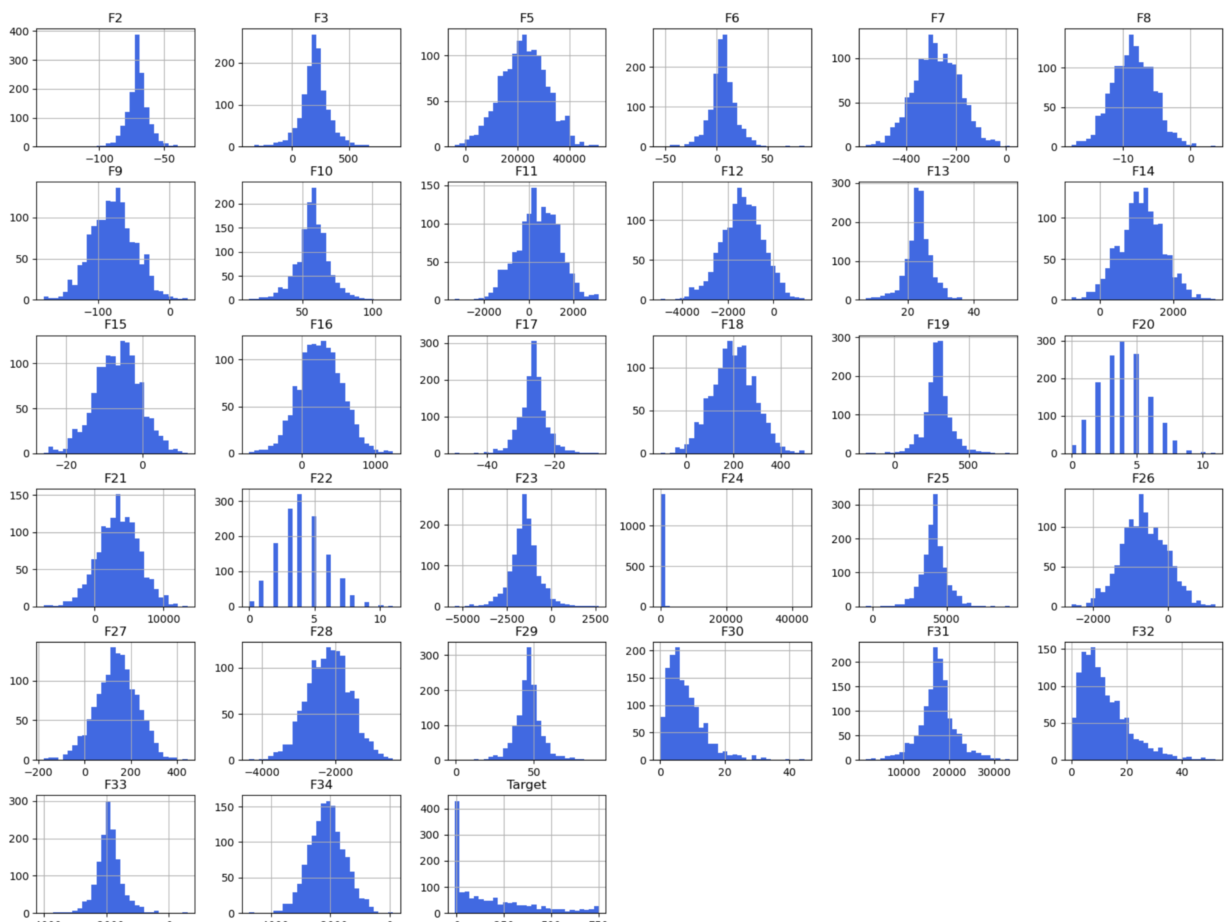
Here, we have a dataset which is larger than the one provided before, which contains many features to predict the blood glucose level, which has been summarized below.

	count	mean	std	min	25%	50%	75%	max
F2	1400.0	-70.299814	8.608240	-142.96	-74.3400	-70.140	-66.1750	-31.90
F3	1400.0	200.416864	132.110502	-376.80	133.8525	200.655	262.1850	887.07
F5	1400.0	22542.739736	8898.976307	-4025.07	16403.3100	22702.650	28688.2650	51163.98
F6	1400.0	6.143893	12.958121	-54.99	-0.3900	6.015	12.4350	85.59
F7	1400.0	-277.502850	91.500104	-559.95	-340.0200	-277.695	-213.0225	15.30
F8	1400.0	-8.500821	3.061879	-17.59	-10.6325	-8.550	-6.3600	3.73
F9	1400.0	-78.977129	30.591408	-176.19	-100.1400	-78.815	-59.0025	24.90
F10	1400.0	57.206186	11.915907	11.67	51.0600	57.360	63.7200	114.78
F11	1400.0	436.422150	932.217719	-3298.53	-180.0000	437.790	1086.3900	3128.43
F12	1400.0	-1371.397993	914.312976	-4954.56	-1963.1250	-1360.875	-753.5700	1356.21
F13	1400.0	23.352943	4.115171	7.31	21.4475	23.360	25.4925	51.05
F14	1400.0	1137.197629	599.765731	-760.46	755.9950	1150.150	1544.2350	3153.16
F15	1400.0	-6.533729	5.881568	-25.94	-10.4200	-6.250	-2.6800	11.78
F16	1400.0	256.788171	291.657468	-709.30	57.3850	253.425	452.6800	1242.35
F17	1400.0	-26.259264	4.139950	-49.48	-28.3125	-26.260	-24.2100	-6.86
F18	1400.0	199.539064	89.517561	-107.91	139.7625	200.820	258.6525	498.87
F19	1400.0	291.895329	89.431742	-190.78	248.2450	290.580	333.2800	775.90
F20	1400.0	4.006429	1.826448	0.00	3.0000	4.000	5.0000	11.00
F21	1400.0	3494.538621	2995.345402	-7382.27	1516.6850	3528.875	5587.8025	13493.44
F22	1400.0	4.077143	1.797001	0.00	3.0000	4.000	5.0000	11.00
F23	1400.0	-1486.052343	827.356749	-5426.02	-1906.0800	-1477.280	-1072.9600	2659.60
F24	1400.0	104.588171	1513.715516	0.00	0.1300	1.000	6.9450	43724.88
F25	1400.0	4147.947543	847.022290	-424.78	3709.3400	4142.250	4580.1800	9220.12
F26	1400.0	-633.753200	601.084635	-2573.38	-1041.6800	-656.190	-220.0300	1278.34
F27	1400.0	139.433871	89.766583	-177.72	81.7425	140.430	199.5000	445.89
F28	1400.0	-2175.604971	587.466383	-4352.42	-2575.8300	-2169.100	-1772.1750	-415.46
F29	1400.0	46.495971	8.620017	-0.58	42.4100	46.540	50.6300	91.38
F30	1400.0	7.910286	5.709544	0.06	3.7800	6.560	10.5650	44.54
F31	1400.0	17456.750529	3994.029442	1728.67	15456.5800	17472.290	19368.2125	33419.90
F32	1400.0	11.974907	8.449665	0.15	5.6925	9.960	16.2975	52.02
F33	1400.0	-1914.526993	434.254039	-4018.60	-2132.6025	-1920.935	-1706.1550	600.93
F34	1400.0	-2111.416129	605.032845	-4751.72	-2519.5400	-2111.570	-1710.2400	121.98
Target	1400.0	181.948593	207.330137	-8.93	-8.9300	114.145	300.2725	750.78

Unlike the other dataset, this doesn't have any blank values, but we have two features with categorical string values which can be treated in many ways to convert it into numerical values. Presenting a correlation plot below,



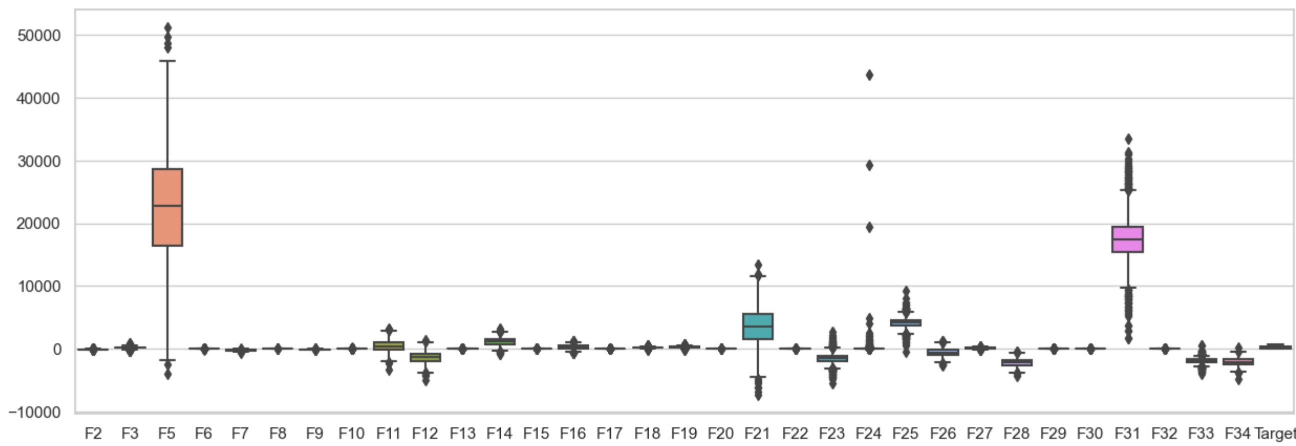
Checking the way of data distributed across every column by an histogram below for analyzing the pattern.



The string categorical value in the columns ‘F1’ and ‘F4’ can be converted into numerical values using OneHotEncoder which basically creates columns with all categories and fills them with ‘1’ with matching categories and ‘0’ if it does not belong to the same one, doing this will greatly improves the accuracy.

	F2	F3	F5	F6	F7	F8	F9	F10	F11	F12	...	Target	0	1	2	3	4	5	6	7	8
0	-48.20	315.96	5570.88	6.27	-312.93	-16.65	-58.22	44.07	2919.36	-1988.43	...	509.59	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
1	-75.20	162.15	6124.62	2.52	-318.66	-11.73	-53.51	70.83	-1171.56	-1734.90	...	-8.93	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	-62.12	346.29	12506.61	-2.85	-257.34	-1.26	-79.76	65.64	-225.09	-1996.32	...	-8.93	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
3	-70.74	280.32	20098.20	9.33	-175.35	-7.31	-90.70	63.72	1604.97	-1589.97	...	22.18	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	-63.00	264.69	13388.76	3.30	-195.51	-6.98	-126.49	56.10	-556.32	-1704.93	...	170.98	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Checking for outliers using the box plot and found some in the column ‘F24’.

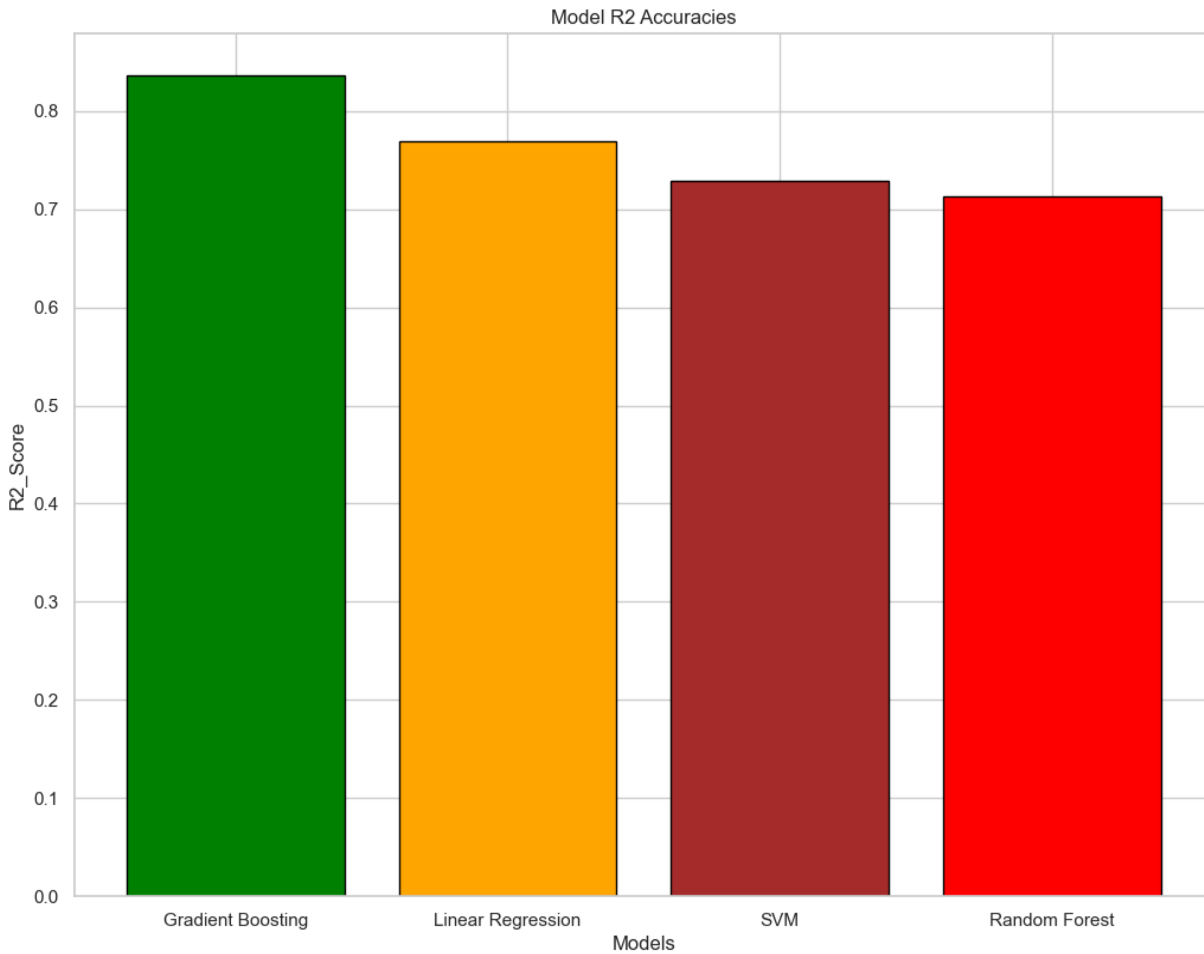


Since there are no missing values, we don’t have to perform imputations. So the following are the models we’re going to use:

1. Linear Regression
2. SVM
3. Random Forest Regressor
4. Gradient Boosting Regressor

The models can be evaluated using many performance metrics like, R2_score, Mean squared error and Mean absolute error, which assess the efficiency of the model and conveys the best one.

Performance Metrics	R2_Score	Mean Squared Error	Mean Absolute Error
Linear Regression	0.769	10801.044	78.91
SVM	0.728	12716.798	81.93
Random Forest	0.713	13446.509	89.99
Gradient Boosting	0.836	7646.689	67.09



The best predicted model was gradient boosting, achieving over 84% of R2_score, followed by the score of Linear Regression. The models have been evaluated using the performance metrics and the values have been tabulated for clarity.

For the next section of the additional comparative study, we used the trained gradient boosting model (best performed) and predicted the patients with risk of developing diabetes using the blood glucose level (diabetic, if exceeds the threshold limit i.e., 125 mg/DL). The approximate numbers are,

Number of patients with higher risk of diabetes: 777

Number of patients with lower risk of diabetes: 523

Conclusion

The report conducted studies to predict the patients with higher risk of developing diabetes with two different dataset and with various constraints. The first one involves the prediction of presence of diabetics while the second, involved in predicting the blood glucose level values. We have used multiple imputation methods to handle the missing values and machine learning models to predict the target features and significantly pulled off better accuracies. This concludes and confirms that machine learning algorithms can be effectively used in hospitals to diagnose the patient's future health condition and can also calculate certain metrics like blood glucose level, thus justifying as a lifesaving tool.