Predicting the Readmission Rate of a Diabetics Patient

Karthick Sharan

Motivation

Hospital readmission is a genuine issue that requires ongoing discussion in order to enhance patient satisfaction and the quality of care while maintaining cost effectiveness. Diabetes is one of the existing chronic diseases across the world. If we know beforehand that a particular diabetic patient has a high chance of readmission we can change the treatment to avoid readmission.

The main aim of this project is to build a machine learning model that can accurately predict patient readmission.

Data

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Diabetes 130-US hospitals for years 1999-2008 Data Set

Download: Data Folder, Data Set Description

Abstract: This data has been prepared to analyze factors related to readmission as well as other outcomes pertaining to patients with diabetes.

Data Set Characteristics:	Multivariate	**Number of Instances:**	100000	**Area:**	Life
Attribute Characteristics:	Integer	**Number of Attributes:**	55	**Date Donated**	2014-05-03
Associated Tasks:	Classification, Clustering	**Missing Values?**	Yes	**Number of Web Hits:**	414623

Information about those patients who met the following criteria:

- Inpatient encounter (also a hospital admission)
- Diabetic encounter, includes any kind of diabetes that entered into system as a diagnosis
- Length of patient stay (at least 1 day and at most 14 days)
- Laboratory tests performed during the encounter
- Medications administered during the encounter

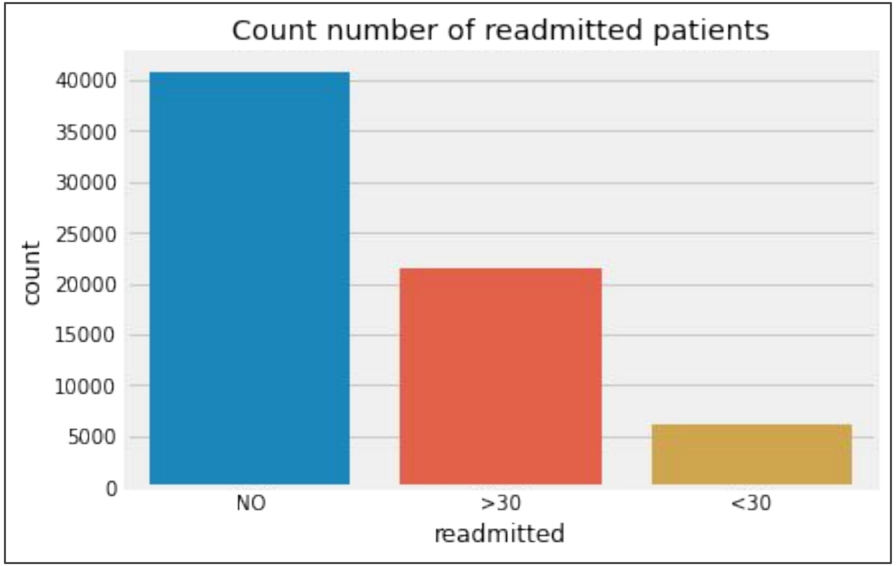
Output: Predicting a diabetic patient will be readmitted to a hospital within a month

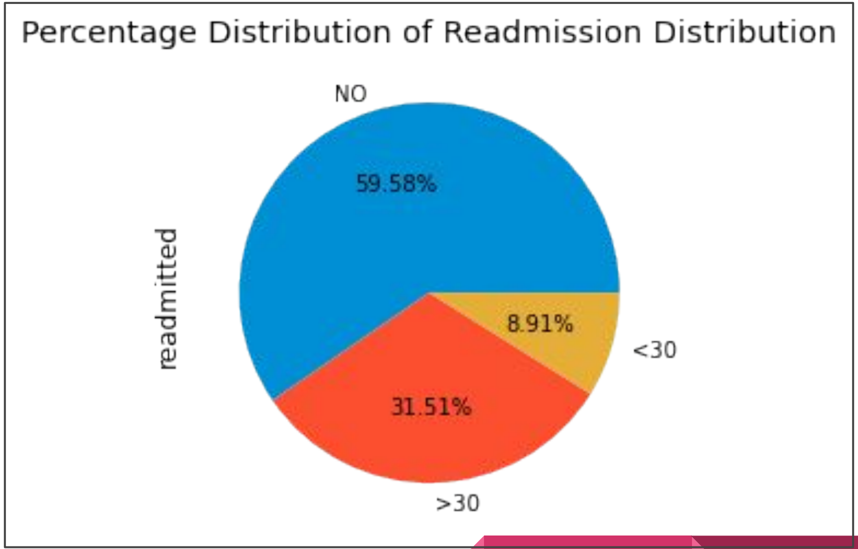
Data Cleaning & EDA

After we cleanse the datasets, here are the key insights to look up for:

- There are 68,358 observations
- The targeted variable is imbalanced
- Dependent variable, “Readmitted”, shows that patients whether could get re-admitted to the hospital within 30 days or not.
- We come up with 3 classifications:
    - **Blue color = the patients readmitted to the hospital have NO records found**
    - **Orange color = the patients readmitted to the hospital past more than 30 days**
    - **Dark yellow color = the patients readmitted to the hospital within 30 days**

Distribution of readmitted patients (Target Variable)





The column ‘readmitted’ tells us if a patient was hospitalized within 30 days, greater than 30 days or not readmitted.

Numerical and Categorical Columns

After Converting variables to the proper data type according to the data dictionary, our dataset has:

- 68,358 observations
- 11 continuous variables
- 35 categorical variables

Numeric Variables:

	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_outpatient	number_emergency	number_inpatient	diag_1	diag_2
1	3	59	0	18	0	0	0	276.0	250.01
2	2	11	5	13	2	0	1	648.0	250.00
3	2	44	1	16	0	0	0	8.0	250.43
4	1	51	0	8	0	0	0	197.0	157.00
5	3	31	6	16	0	0	0	414.0	411.00
...	...	...	...	...	...	...	...	...	...
101754	9	50	2	33	0	0	0	574.0	574.00
101755	14	73	6	26	0	1	0	592.0	599.00
101756	2	46	6	17	1	1	1	996.0	585.00
101758	5	76	1	22	0	1	0	292.0	8.00
101765	6	13	3	3	0	0	0	530.0	530.00

68358 rows × 11 columns

Categorical Variables:

	race	gender	age	admission_type_id	discharge_disposition_id
1	Caucasian	Female	[10-20)	1	1
2	AfricanAmerican	Female	[20-30)	1	1
3	Caucasian	Male	[30-40)	1	1
4	Caucasian	Male	[40-50)	1	1
5	Caucasian	Male	[50-60)	2	1
...	...	...	...	...	...
101754	Caucasian	Female	[70-80)	1	1
101755	Other	Female	[40-50)	1	1
101756	Other	Female	[60-70)	1	1
101758	Caucasian	Female	[80-90)	1	1
101765	Caucasian	Male	[70-80)	1	1

68358 rows × 35 columns

Reducing Unique values in Categorical variables

```
high_frequency = ['InternalMedicine', 'Family/GeneralPrac
                  'Emergency/Trauma', 'Urology','Obstetric

low_frequency = ['Surgery-PlasticwithinHeadandNeck','Psyc
                 'Neurophysiology','Resident','Pediatrics-
                 'Pediatrics-Pulmonology','Surgery-Pediatr
                 'Endocrinology-Metabolism','PhysicianNotF
                 'Surgery-Maxillofacial','Rheumatology','A

pediatrics = ['Pediatrics','Pediatrics-CriticalCare','Ped
              'Pediatrics-Neurology','Pediatrics-Pulmono

psychic = ['Psychiatry-Addictive', 'Psychology', 'Psychia


neurology = ['Neurology', 'Surgery-Neuro',  'Pediatrics-N


surgery = ['Surgeon', 'Surgery-Cardiovascular', \
           'Surgery-Cardiovascular/Thoracic', 'Surgery-Col
              'Surgery-Plastic', 'Surgery-PlasticwithinHea
              'Surgery-Vascular', 'SurgicalSpecialty', 'Po

others = ['Endocrinology','Gastroenterology','Gynecology'
          'Oncology','Ophthalmology','Otolaryngology','P


missing = ['?']
```

```
colMedical = []

for val in df['medical_specialty'] :
    if val in pediatrics :
        colMedical.append('pediatrics')
    elif val in psychic :
        colMedical.append('psychic')
    elif val in neurology :
        colMedical.append('neurology')
    elif val in surgery :
        colMedical.append('surgery')
    elif val in high_frequency :
        colMedical.append('high_freq')
    elif val in low_frequency :
        colMedical.append('low_freq')
    elif val in others :
        colMedical.append('others')
    elif val in missing :
        colMedical.append('missing')
    else:
        colMedical.append('?')

df['medical_specialty'] = colMedical
```

Before proceeding with our analysis, for variables having huge number of classes, we've classified them into general categories.

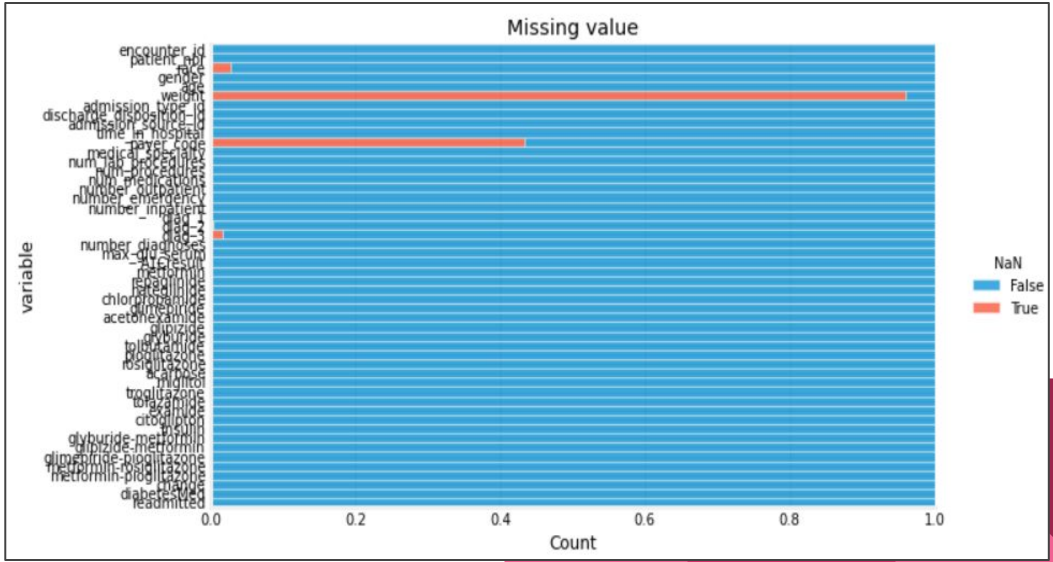
For this one column we reduced the unique values from 73 to just 9.

Handling Missing Values

Columns with more than 40% missing data was dropped (2 columns)

‘Race’ which is categorical has 2% missing data, for this the missing rows were dropped.

	Column Name	Missing Values	Missing Percentage
0	encounter_id	0	0
1	patient_nbr	0	0
2	race	1948	2
3	gender	0	0
4	age	0	0
5	weight	68665	96
6	admission_type_id	0	0
7	discharge_disposition_id	0	0
8	admission_source_id	0	0
9	time_in_hospital	0	0
10	payer_code	31043	43



Handling Missing Values (Imputation)

For Features ‘Diag1’, ‘Diag2’ and ‘Diag3’ we have non-numeric values such as ‘E27’, ‘V55’ etc. in addition to missing values. These values didn’t convey any specific meaning so it was treated as missing and imputation was performed.

KNN-Imputation was used to impute the missing values (after Scaling the numeric features and encoding the categorical ones).

```
# Converting diag_1, diag_2 and diag_3 to numeric
df['diag_1'] = df['diag_1'].apply(lambda x: np.nan if (x[0]=='V' or x[0]=='E') else x)
df['diag_1'] = df['diag_1'].astype('float')
df['diag_2'] = df['diag_2'].apply(lambda x: np.nan if (x[0]=='V' or x[0]=='E') else x)
df['diag_2'] = df['diag_2'].astype('float')
df['diag_3'] = df['diag_3'].apply(lambda x: np.nan if (x[0]=='V' or x[0]=='E') else x)
df['diag_3'] = df['diag_3'].astype('float')
```

```
df[['diag_1','diag_2','diag_3']].isnull().sum()
```

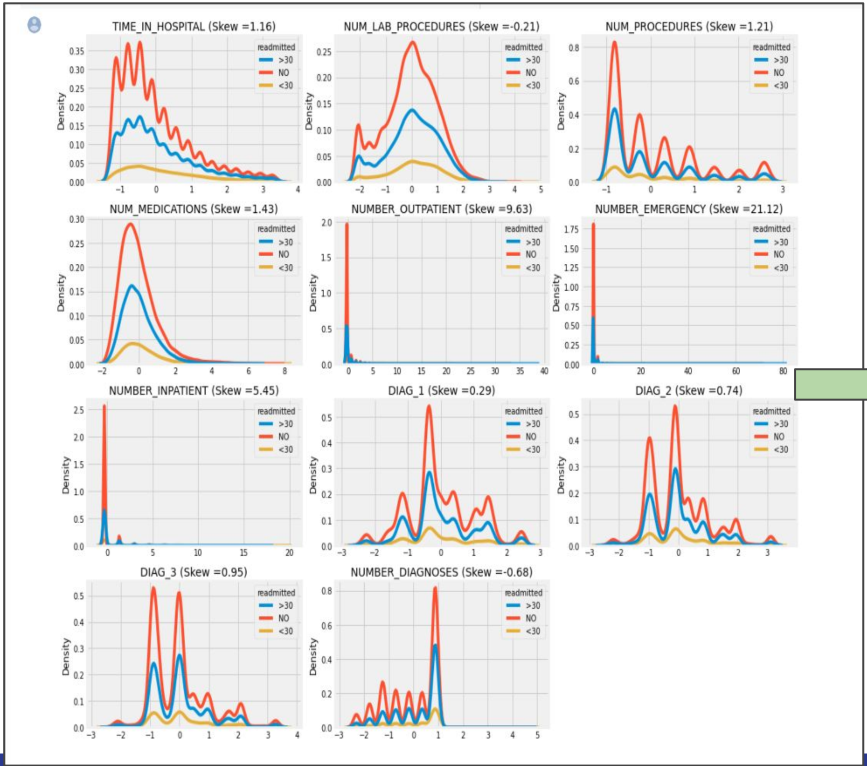
```
diag_1     895
diag_2    1735
diag_3    3469
dtype: int64
```

```
# knn imputation
imputer = KNNImputer(n_neighbors=5)
df_num_imp = pd.DataFrame(imputer.fit_t
df_num_imp.isnull().sum()
```

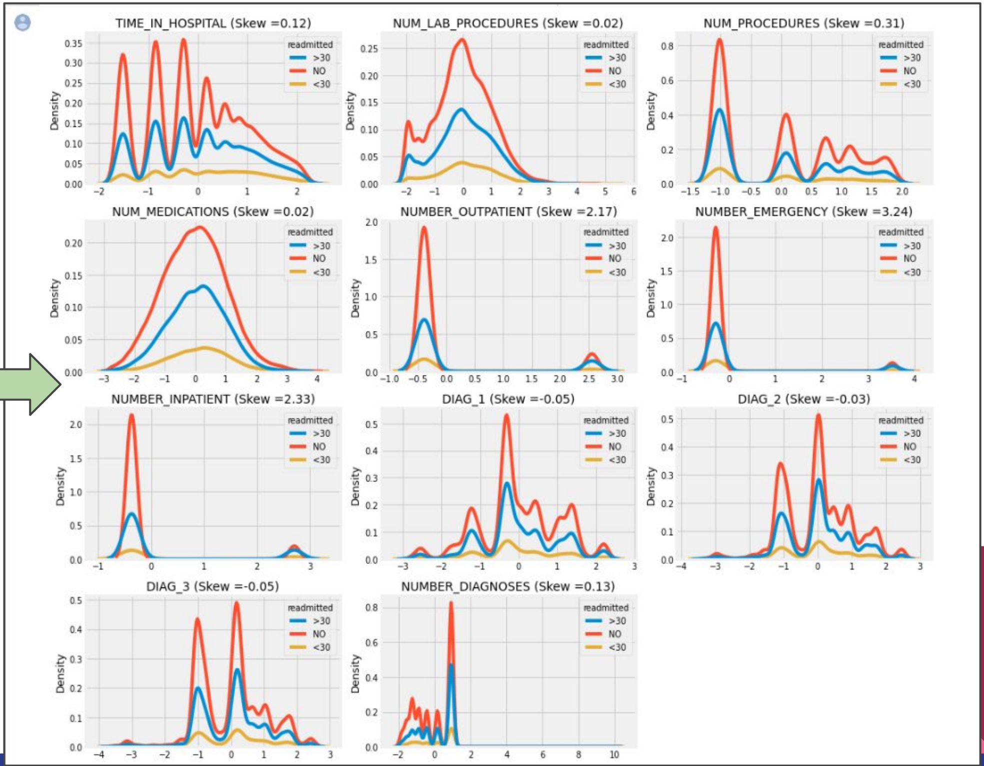
```
time_in_hospital      0
num_lab_procedures    0
num_procedures        0
num_medications       0
number_outpatient     0
number_emergency      0
number_inpatient      0
diag_1                0
diag_2                0
diag_3                0
number_diagnoses      0
dtype: int64
```

EDA for Numerical Variables

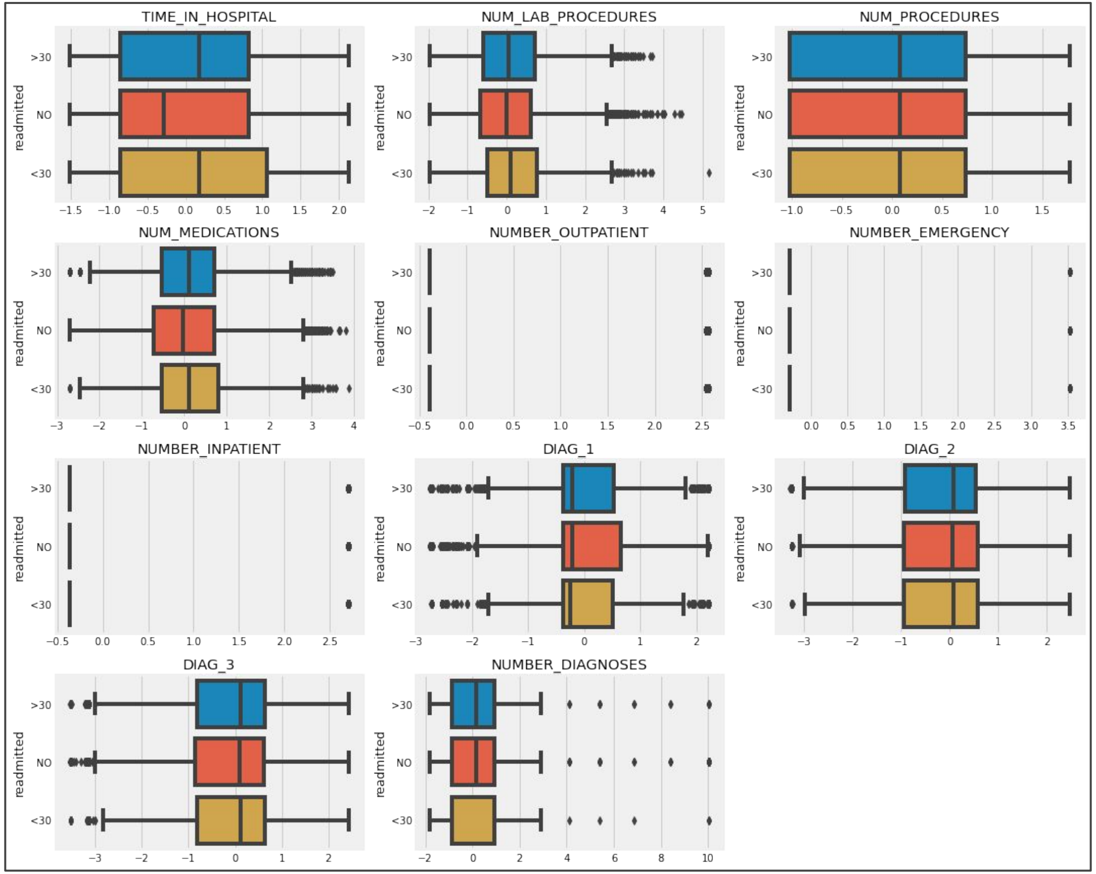
Skewness Before Power transformation



Skewness After transformation (yeo-johnson)



Numerical Features Against Target

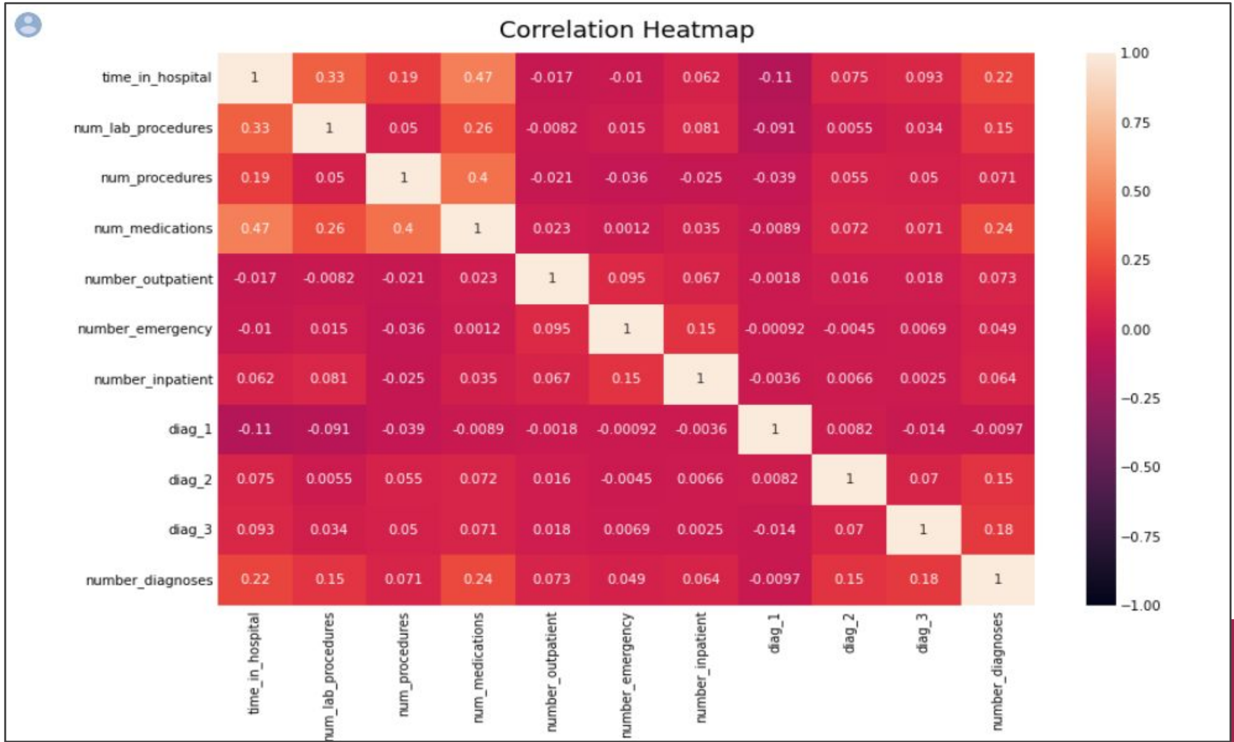


There doesn’t seem to be a significant difference in the IQR of the 3 boxes in the plots.

Time_in_hospital and Num_lab_procedures seem to slightly affect the target which doesn’t seem to have much impact.

Treating Multicollinearity and Feature Extraction

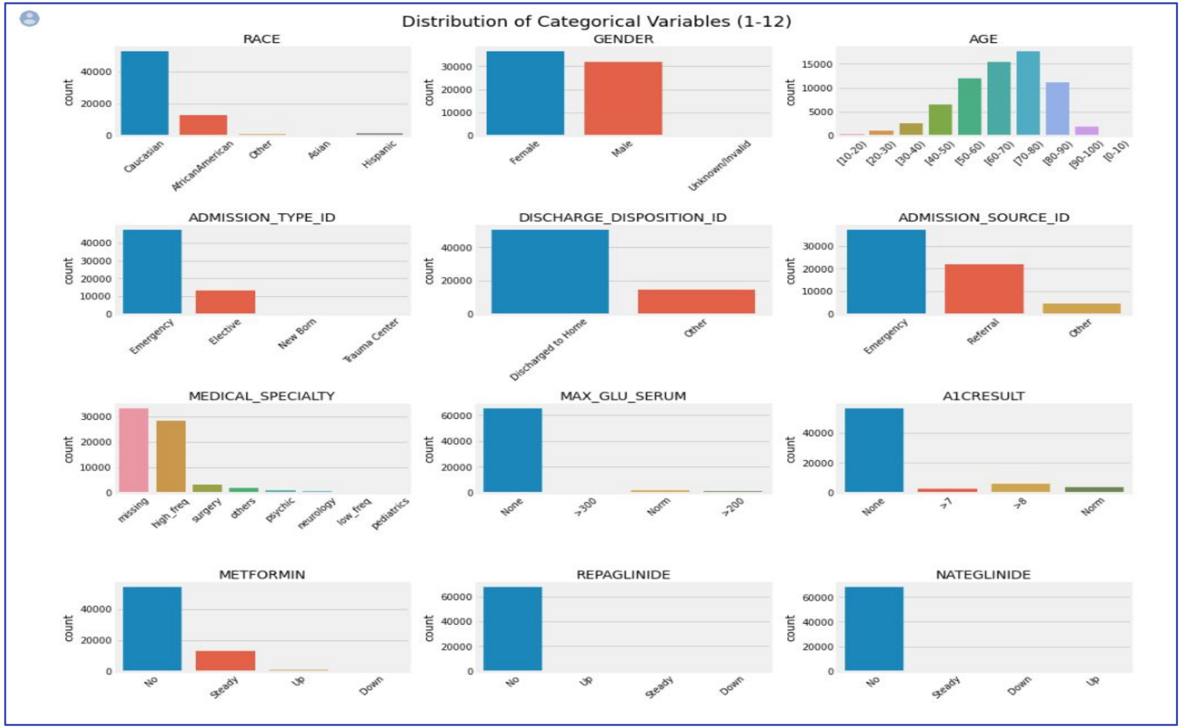
	feature	VIF
0	time_in_hospital	1.399188
1	num_lab_procedures	1.162083
2	num_procedures	1.204304
3	num_medications	1.539338
4	number_outpatient	1.019639
5	number_emergency	1.032789
6	number_inpatient	1.036174
7	diag_1	1.020271
8	diag_2	1.028674
9	diag_3	1.038367
10	number_diagnoses	1.138463

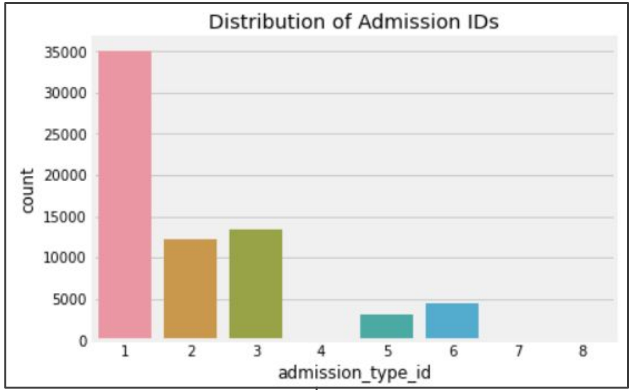


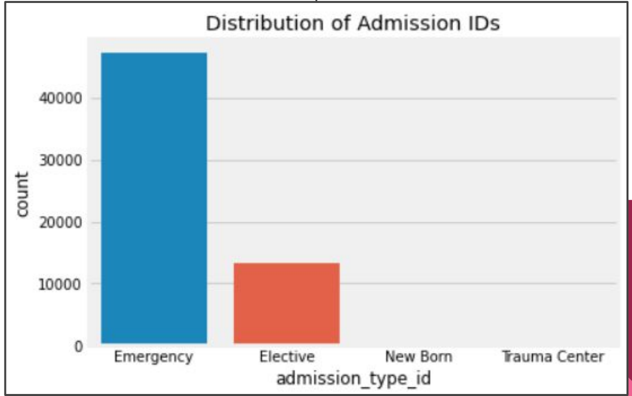
No multicollinearity present since VIF is <2 for all numeric features

EDA Categorical Variables

Remapping sub-categories according to Data dictionary, and exploring the distribution of categorical variables.

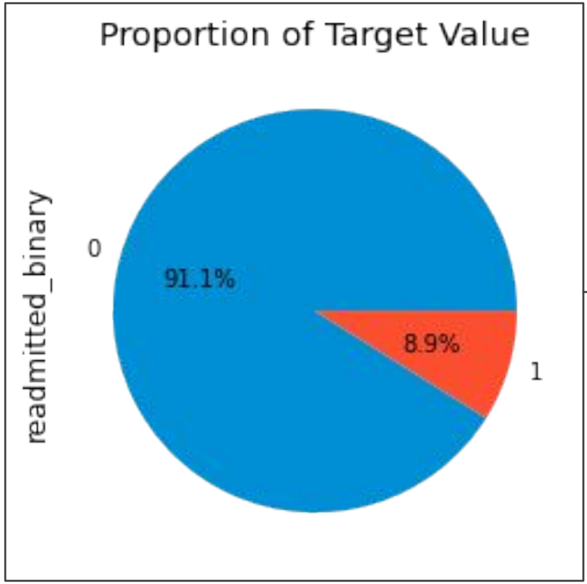






Re-Mapping the Target variable into Binary

readmitted / age	<30	>30	NO
[0-10)	1	12	51
[10-20)	20	98	222
[20-30)	76	241	675
[30-40)	178	656	1671
[40-50)	482	1875	4141
[50-60)	853	3636	7471
[60-70)	1381	4822	9170
[70-80)	1779	5973	9836
[80-90)	1160	3761	6281
[90-100)	160	465	1211

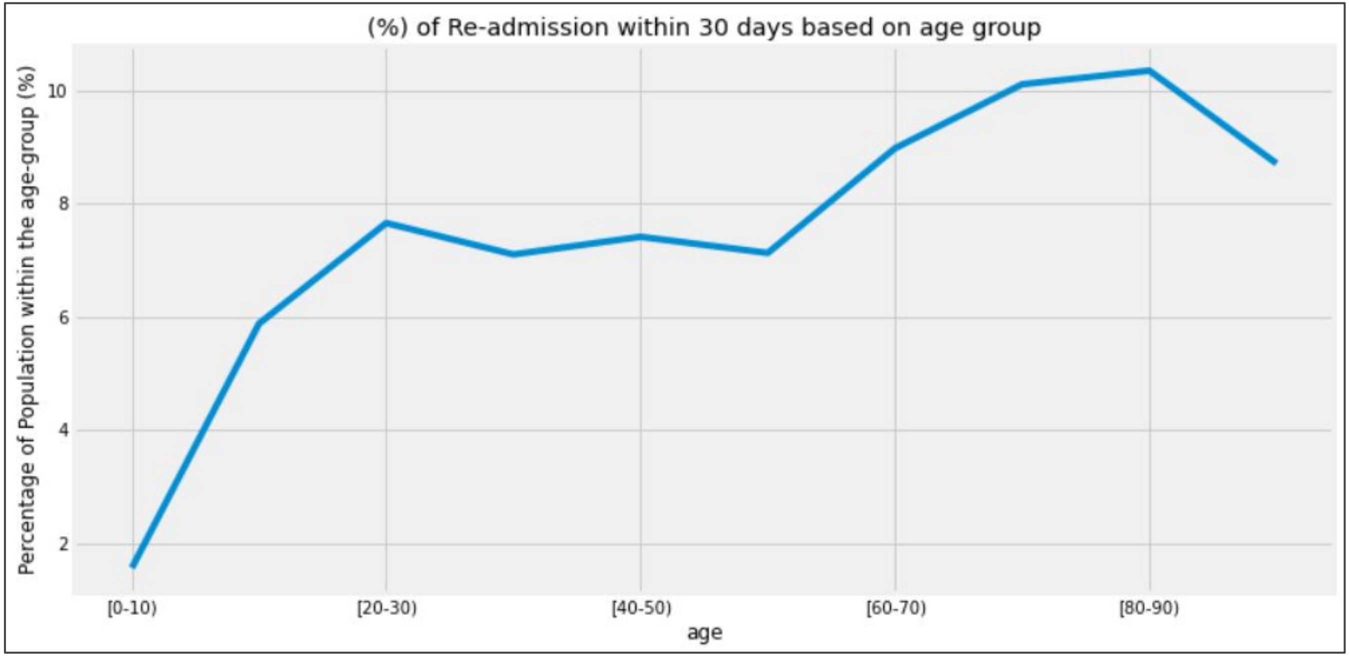


0: patients are readmitted to the hospital past 30 days or no records

1: patients are readmitted to the hospital within 30 days

readmitted_binary / age	0	1
[0-10)	63	1
[10-20)	320	20
[20-30)	916	76
[30-40)	2327	178
[40-50)	6016	482
[50-60)	11107	853
[60-70)	13992	1381
[70-80)	15809	1779
[80-90)	10042	1160
[90-100)	1676	160

Age group vs Target variable (Readmitted)



readmitted_binary	0	1
age		
[0-10)	98.437500	1.562500
[10-20)	94.117647	5.882353
[20-30)	92.338710	7.661290
[30-40)	92.894212	7.105788
[40-50)	92.582333	7.417667
[50-60)	92.867893	7.132107
[60-70)	91.016718	8.983282
[70-80)	89.885149	10.114851
[80-90)	89.644706	10.355294
[90-100)	91.285403	8.714597

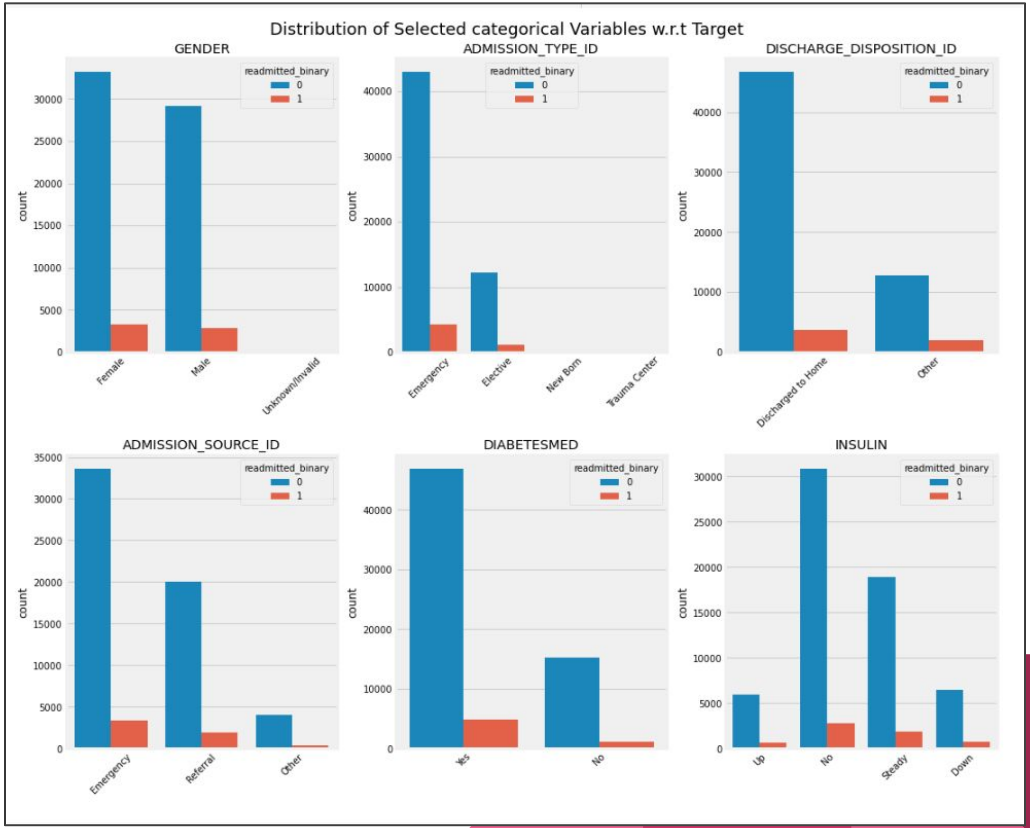
The above plot illustrates the (%) of population within the age group who got readmitted within 30 days, and there seems to be a strong correlation.

Categorical Variables vs Target

- Female has a higher proportion of population, but the count of readmittance of Male seems be equal to that of female

- Similarly Discharge_disposition_id may also impact the re-admittance rate

- For other variables though not evidently visible, there might still be some relation with the target



Proportion of Target in Categorical variables

```
for i in cols:
    print('Varibale:',i,'\n')
    print(pd.crosstab(df_cat[i],df_cat['readmitted_binary
    print("-----------------------------------------------
```

```
Varibale: gender

readmitted_binary              0          1
gender
Female                 91.023600   8.976400
Male                   91.167716   8.832284
Unknown/Invalid       100.000000   0.000000
----------------------------------------------------------

Varibale: admission_type_id

readmitted_binary              0          1
admission_type_id
Elective               91.662918   8.337082
Emergency              90.985152   9.014848
New Born               88.888889  11.111111
Trauma Center         100.000000   0.000000
----------------------------------------------------------
```

```
Varibale: discharge_disposition_id

readmitted_binary                 0          1
discharge_disposition_id
Discharged to Home        92.540215   7.459785
Other                     86.233011  13.766989
-----------------------------------------------------

Varibale: admission_source_id

readmitted_binary              0          1
admission_source_id
Emergency              90.859736   9.140264
Other                  91.677882   8.322118
Referral               91.329717   8.670283
-----------------------------------------------------

Varibale: diabetesMed

readmitted_binary            0          1
diabetesMed
No                   92.570463   7.429537
Yes                  90.617699   9.382301
-----------------------------------------------------

Varibale: insulin

readmitted_binary            0          1
insulin
Down                 89.460546  10.539454
No                   91.794506   8.205494
Steady               90.784182   9.215818
Up                   90.261211   9.738789
-----------------------------------------------------
```

Data Preprocessing before modelling

Power Transformation(Yeo-Johnson) was used to reduce the skew in numeric data, Scaling was performed using Standard Scalar.

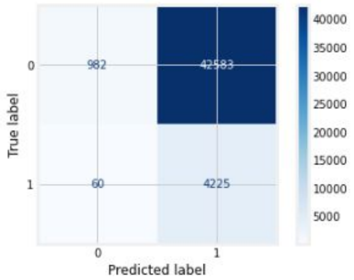
Categorical variables were dummy encoded (with drop_first)

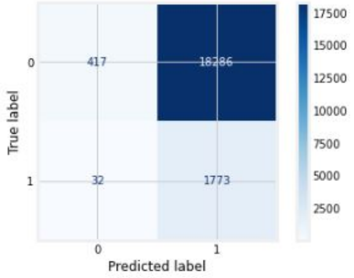
```
Shape of Final Dataframe: (68358, 92)
```

num_medications	number_outpatient	number_emergency	number_inpatient	diag_1	diag_2	diag_3	...	insulin_No	insulin_Steady	insulin_Up	metf
0.489262	-0.390600	-0.283670	-0.369736	-1.114593	-1.149007	-1.001435	...	0	0	1	
-0.192788	2.575451	-0.283670	2.702722	0.775035	-1.149085	1.112344	...	1	0	0	
0.244787	-0.390600	-0.283670	-0.369736	-2.724181	-1.145730	0.114383	...	0	0	1	
-1.103596	-0.390600	-0.283670	-0.369736	-1.572916	-1.910545	-1.044846	...	0	1	0	
0.244787	-0.390600	-0.283670	-0.369736	-0.354894	-0.017281	-1.044846	...	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	
1.744963	-0.390600	-0.283670	-0.369736	0.438976	0.844384	-1.044672	...	0	1	0	
1.243959	-0.390600	3.525205	-0.369736	0.522150	0.958256	0.741328	...	0	0	1	
0.371156	2.544766	3.525205	2.702722	2.212598	0.894925	0.114383	...	0	1	0	
0.900432	-0.390600	3.525205	-0.369736	-1.023676	-3.264887	-0.594717	...	0	0	1	
-2.216890	-0.390600	-0.283670	-0.369736	0.231118	0.634605	1.796355	...	1	0	0	

Fitting Base Model (Naive Bayes)

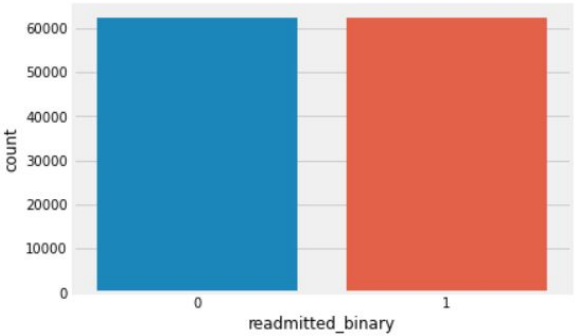
```
Gaussian Naive Bayes Performance:
Train Acurracy: 0.10881922675026123
Training error is: 0.8911807732497388
Test Acurracy: 0.10678759508484494
Test error is: 0.8932124049151551
              precision    recall  f1-score   support

           0       0.93      0.02      0.04     18703
           1       0.09      0.98      0.16      1805

    accuracy                           0.11     20508
   macro avg       0.51      0.50      0.10     20508
weighted avg       0.85      0.11      0.05     20508
```



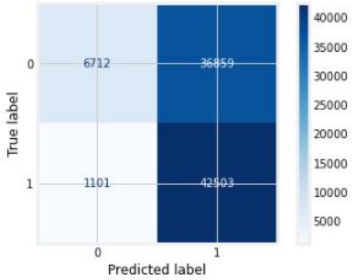


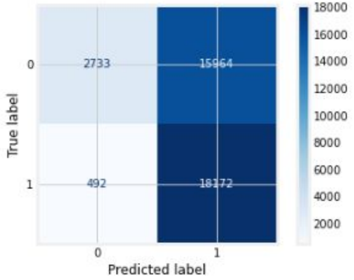
Using Naive Bayes as Baseline model

Applying SMOTE to balance the target class.

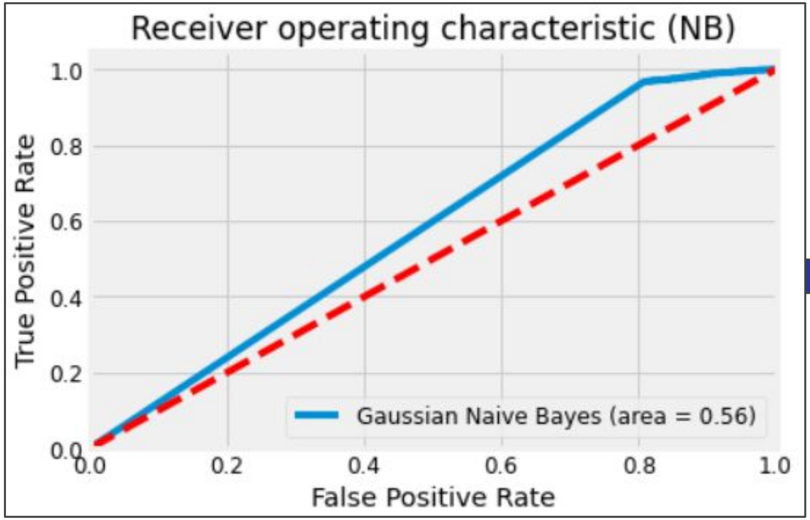


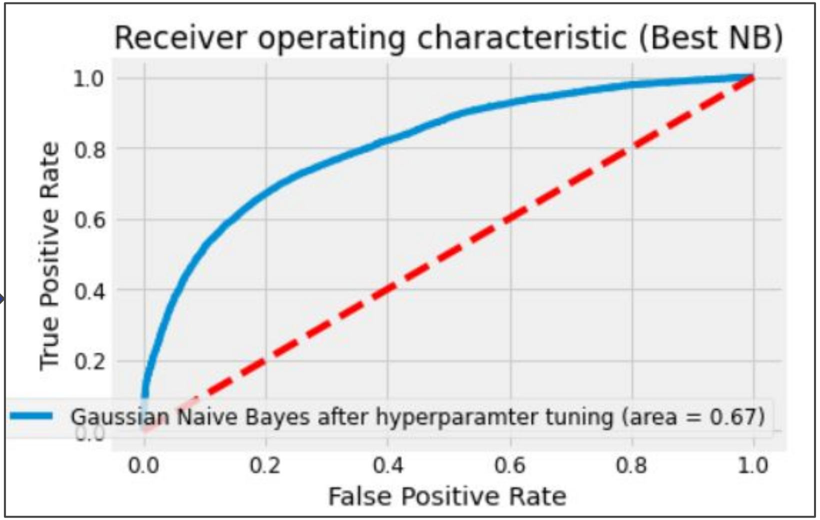

```
Gaussian Naive Bayes Performance:
Train Acurracy: 0.5645540579294522
Training error is: 0.43544594207054776
Test Acurracy: 0.559540697518803
Test error is: 0.440459302481197
              precision    recall  f1-score   support

           0       0.85      0.15      0.25     18697
           1       0.53      0.97      0.69     18664

    accuracy                           0.56     37361
   macro avg       0.69      0.56      0.47     37361
weighted avg       0.69      0.56      0.47     37361
```





Naive Bayes - Hyperparameter Tuning





AUC = 0.56 Before hyperparameter tuning

ROC AUC = 0.67 after hyperparameter tuning using gridsearchcv (cv=5)

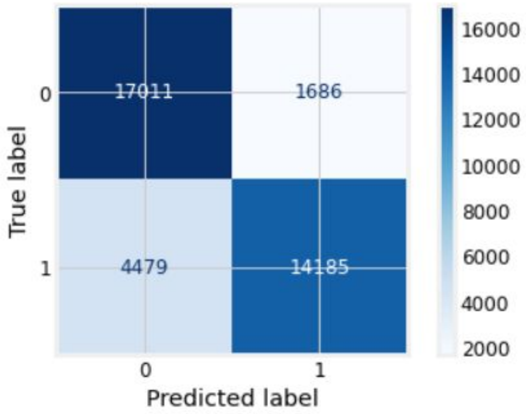
```
The best parameters are:  {'var_smoothing': 0.012742749857031341}
```

Logistic Regression Model

Before Hyperparameter Tuning

	precision	recall	f1-score	support
0	0.79	0.91	0.85	18697
1	0.89	0.76	0.82	18664
accuracy			0.83	37361
macro avg	0.84	0.83	0.83	37361
weighted avg	0.84	0.83	0.83	37361

Confusion Matrix for test data:

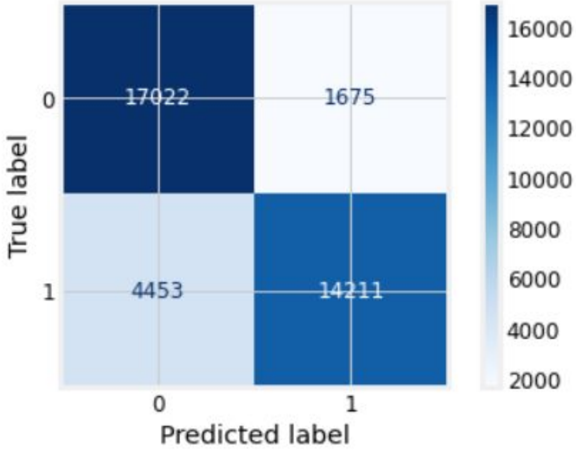


GridsearchCV
(cv=5)

```python
param_grid = {'solver':['newton-cg', 'lbfgs'],
              'C': [0.5,1,5],
              'penalty':['l1', 'l2', 'elasticnet']}
```

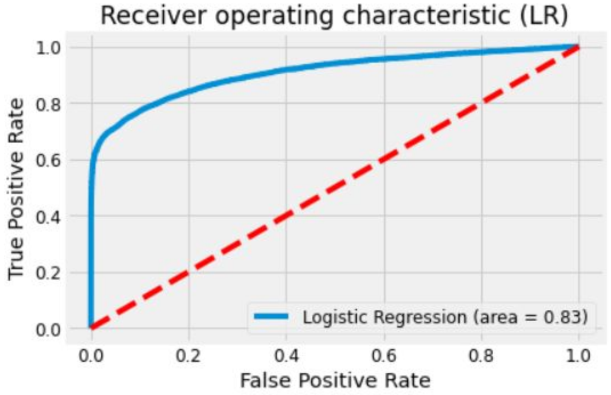
After Hyperparameter Tuning

	precision	recall	f1-score	support
0	0.79	0.91	0.85	18697
1	0.89	0.76	0.82	18664
accuracy			0.84	37361
macro avg	0.84	0.84	0.84	37361
weighted avg	0.84	0.84	0.84	37361

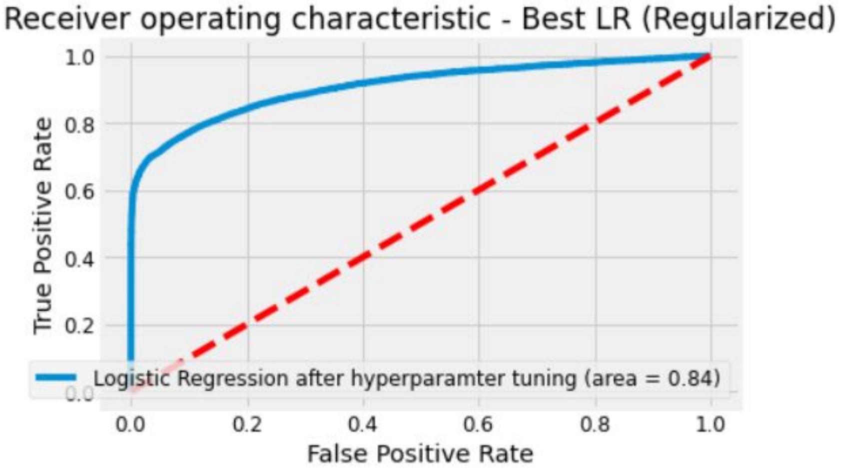


Logistic Regression (Tuning Hyperparameter/Regularization)

ROC AUC Score before hyperparameter tuning: 0.8349221970508505



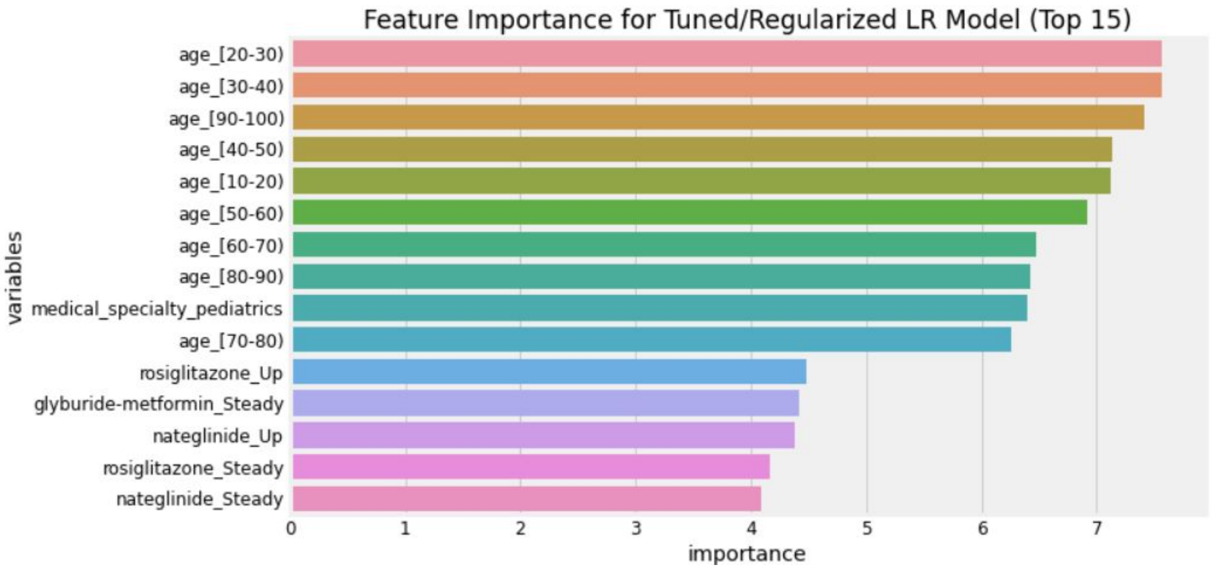
ROC AUC Score after tuning Hyperparameters/Regularization: 0.8359128899655688

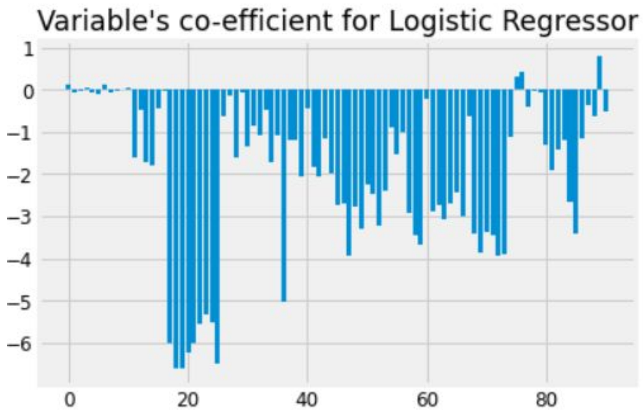



```
The best parameters:  {'C': 5, 'penalty': 'l2', 'solver': 'newton-cg'}
```

Feature Importance (Logistic Regression)

The below plot depicts the 15 most important features used by the LR model for prediction, and the values of the coefficients for the 92 variables.



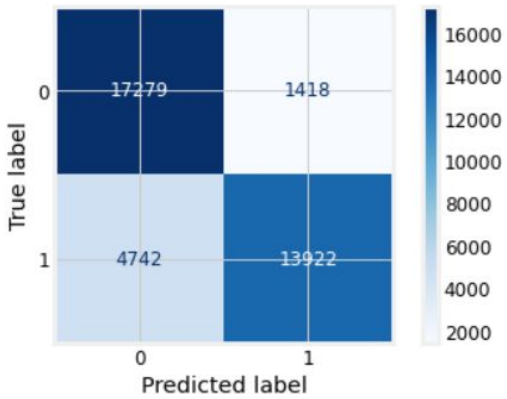


SVM (Base vs Tuned Model)

Classification Report:

	precision	recall	f1-score	support
0	0.78	0.92	0.85	18697
1	0.91	0.75	0.82	18664
accuracy			0.84	37361
macro avg	0.85	0.84	0.83	37361
weighted avg	0.85	0.84	0.83	37361

Confusion Matrix SVM (Test data):



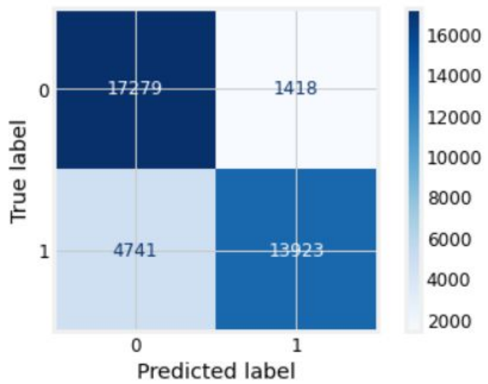
GridsearchCV
(cv=5)

```
param_grid = {'penalty': ['l1','l2'],
              'C': [0.5, 1, 10, 100, 500]}
```

Classification Report:

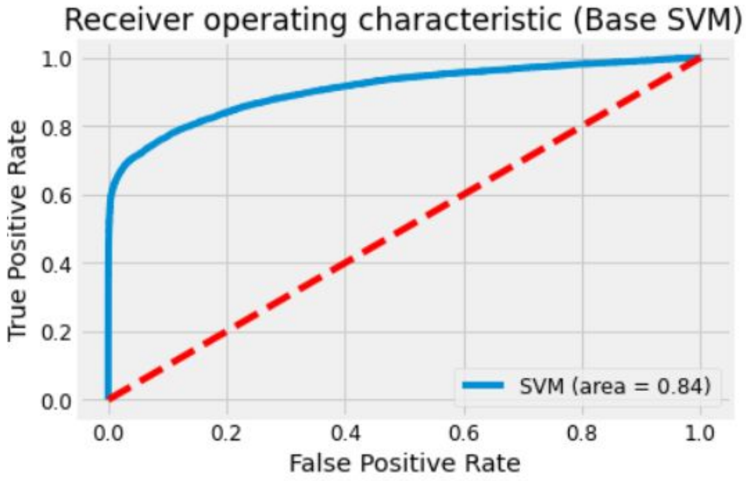
	precision	recall	f1-score	support
0	0.78	0.92	0.85	18697
1	0.91	0.75	0.82	18664
accuracy			0.84	37361
macro avg	0.85	0.84	0.83	37361
weighted avg	0.85	0.84	0.83	37361

Confusion Matrix SVM (Test data):

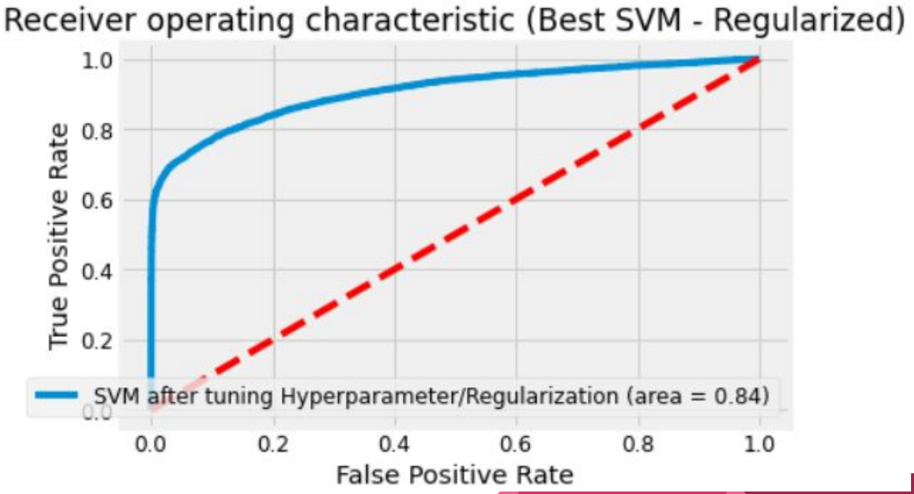


SVM (tuning Hyperparameter/Regularization)

ROC AUC Score before hyperparameter tuning (SVM):
0.8350434728475297



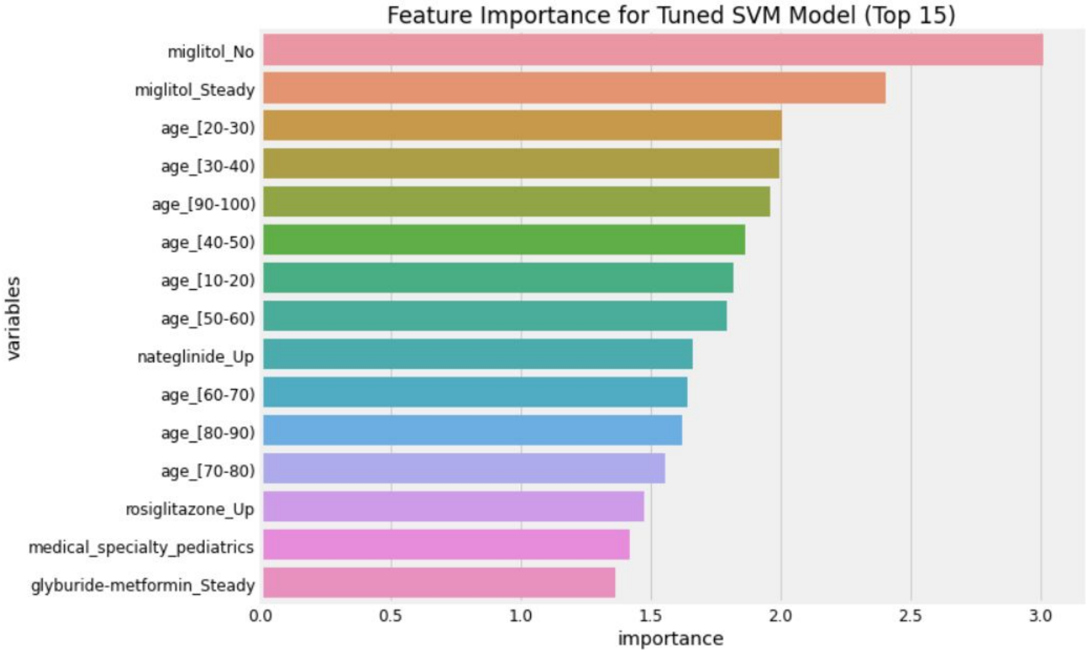
ROC AUC Score of SVM model after tuning Hyperparameters & Regularization:
0.8350702623888927

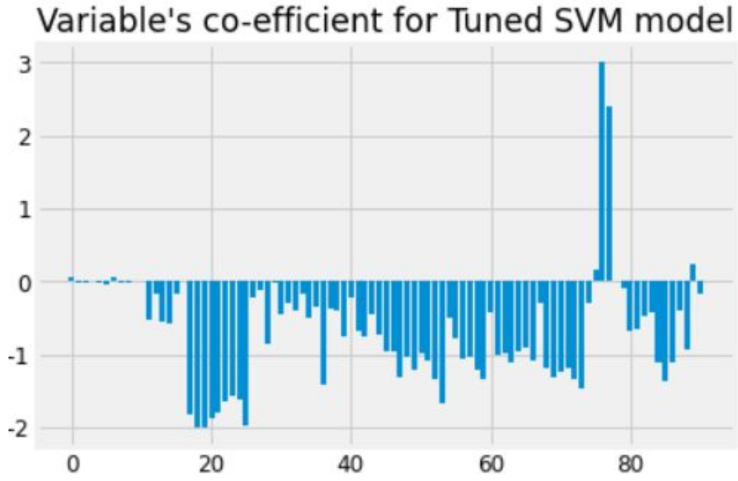


The best parameters: {'C': 1, 'penalty': 'l2'}

SVM - Feature Importance

The below plot depicts the 15 most important features used by the SVM model for prediction, and the values of the coefficients for the 92 variables.





Random Forest Classifier

Classification Report:

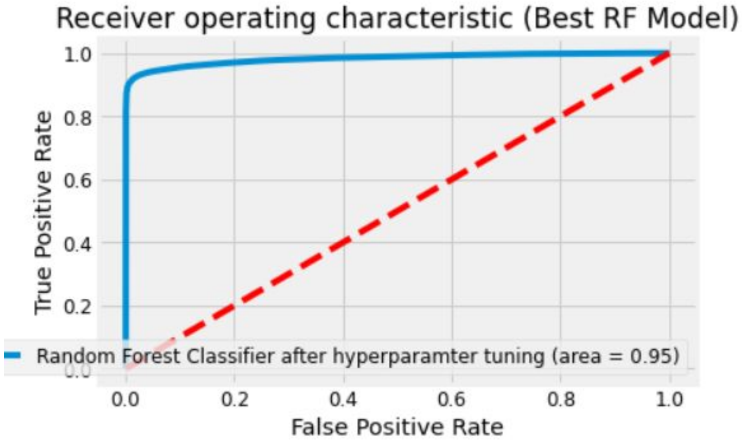
	precision	recall	f1-score	support
0	0.92	0.99	0.95	18697
1	0.99	0.92	0.95	18664
accuracy			0.95	37361
macro avg	0.95	0.95	0.95	37361
weighted avg	0.95	0.95	0.95	37361

GridsearchCV

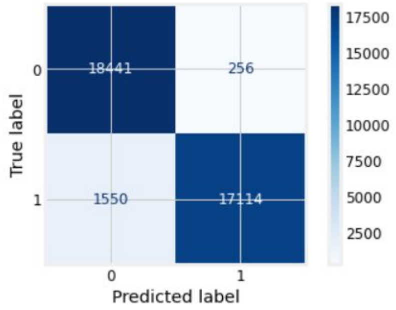
Classification Report:

	precision	recall	f1-score	support
0	0.92	0.99	0.95	18697
1	0.99	0.92	0.95	18664
accuracy			0.95	37361
macro avg	0.95	0.95	0.95	37361
weighted avg	0.95	0.95	0.95	37361

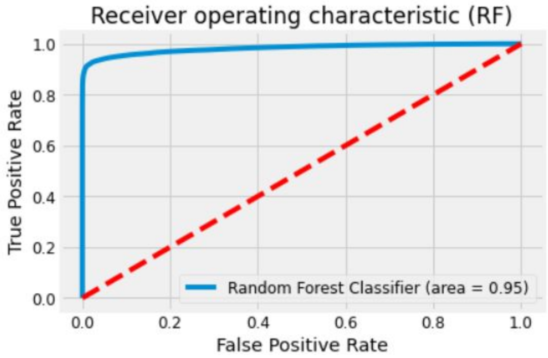
ROC AUC Score after hyperparameter tuning (Best RF): 0.952486559751432



Confusion Matrix for test data:



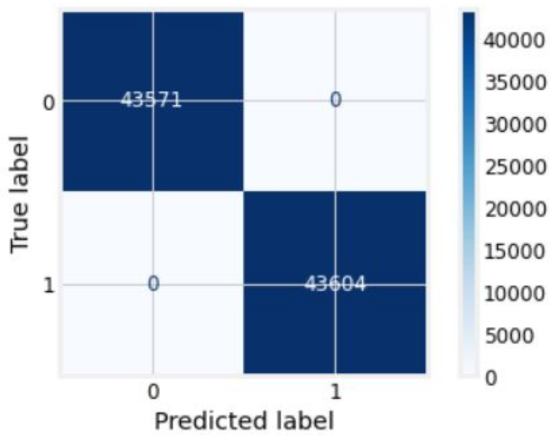
ROC AUC Score before tuning (RF Model): 0.9516301928095031



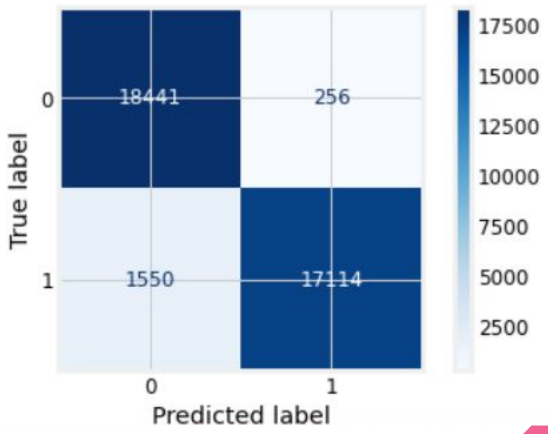
Random Forest (Overfitting?)

- We can observe from the training data's confusion matrix that the RF model is actually overfitting.
- One possible reason for this might be the minority oversampling which we did earlier for our target variable.
- So to avoid overfitting we'll generalize the RF model by tuning it's max_depth in addition to the earlier tuned hyperparameters.

Confusion Matrix for training data:



Confusion Matrix for test data:

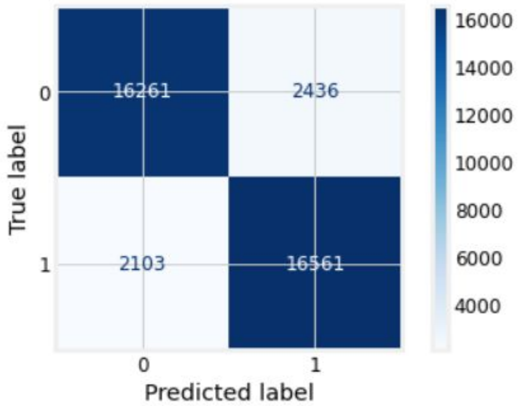


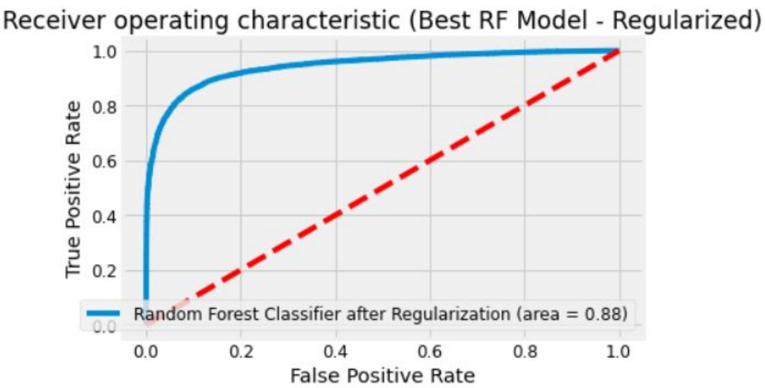
Random Forest Hyperparameter tuning & Regularization

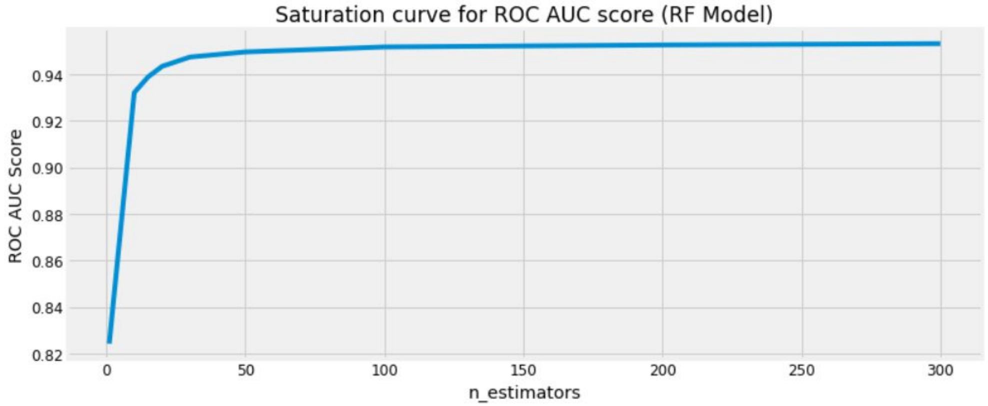
Classification Report:

	precision	recall	f1-score	support
0	0.89	0.87	0.88	18697
1	0.87	0.89	0.88	18664
accuracy			0.88	37361
macro avg	0.88	0.88	0.88	37361
weighted avg	0.88	0.88	0.88	37361

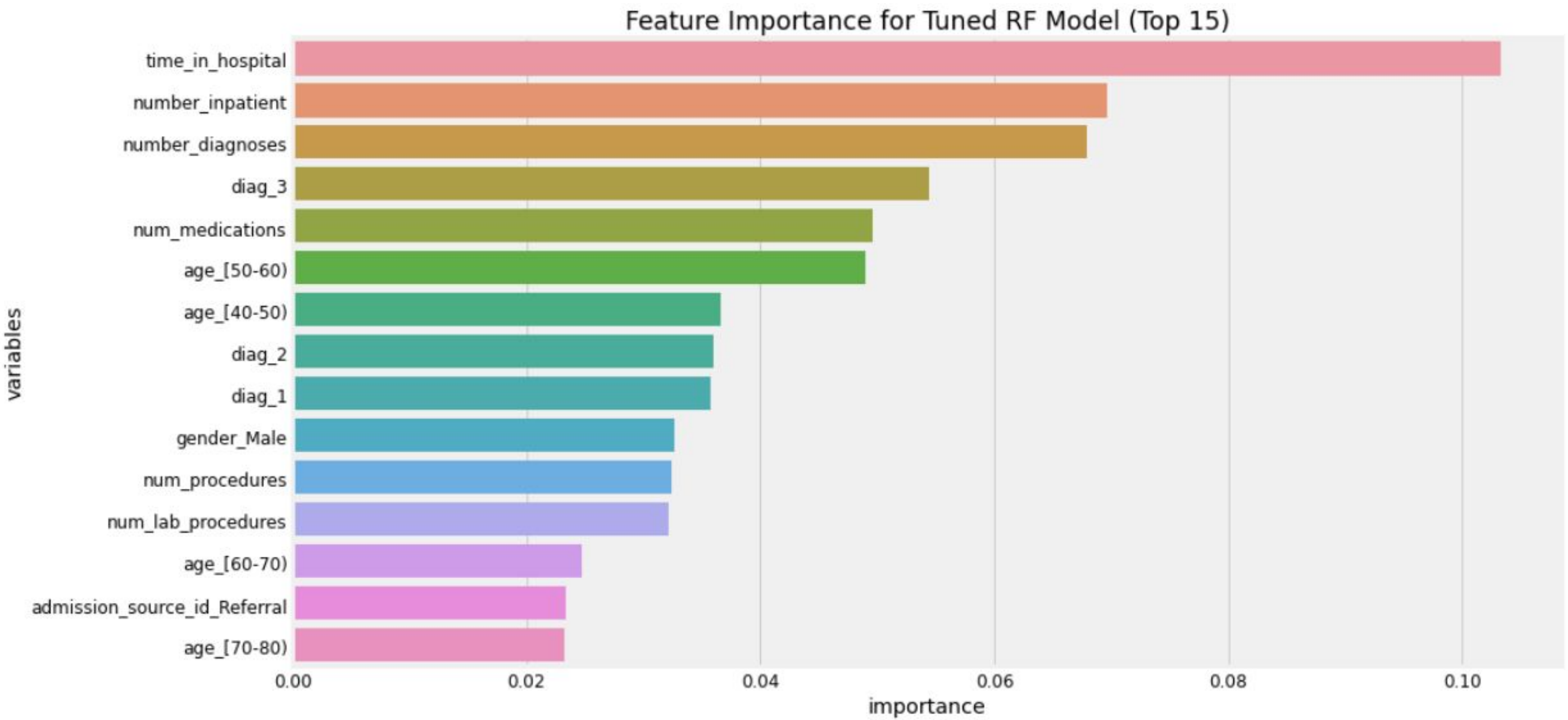
ROC AUC Score after Regularization (Best RF): 0.8785174537422552







Random Forest Feature Importance



Feature Importance for the 3 models

```
top_features = pd.DataFrame({'Top Features (LR)':imp_lr_series.values,
                             'Top Features (SVM)':imp_svm_series.values,
                             'Top Features (RF)':imp_rf_series.values}).head(15)
top_features.index = [i for i in range(1,16)]
top_features
```

	Top Features (LR)	Top Features (SVM)	Top Features (RF)
1	age_[20-30)	miglitol_No	time_in_hospital
2	age_[30-40)	miglitol_Steady	number_inpatient
3	age_[90-100)	age_[20-30)	number_diagnoses
4	age_[40-50)	age_[30-40)	diag_3
5	age_[10-20)	age_[90-100)	num_medications
6	age_[50-60)	age_[40-50)	age_[50-60)
7	age_[60-70)	age_[10-20)	age_[40-50)
8	age_[80-90)	age_[50-60)	diag_2
9	medical_specialty_pediatrics	nateglinide_Up	diag_1
10	age_[70-80)	age_[60-70)	gender_Male
11	rosiglitazone_Up	age_[80-90)	num_procedures
12	glyburide-metformin_Steady	age_[70-80)	num_lab_procedures
13	nateglinide_Up	rosiglitazone_Up	age_[60-70)
14	rosiglitazone_Steady	medical_specialty_pediatrics	admission_source_id_Referral
15	nateglinide_Steady	glyburide-metformin_Steady	age_[70-80)

- This table depicts the top 15 important features for Logistic Regression, SVM and Random Forest model respectively.
- We can observe that while the order of important features varies for each model, there are still quite a few features which are consistent across all the models.

Feature Selection

We've used Recursive Feature Elimination to select the to 45 features to reduce modelling time and complexity.

```
# Using recursive feature elimination
n = 10
best_lr = LogisticRegression(C=1, solver='lbfgs')
rfe = RFE(best_lr, n_features_to_select=n)
rfe.fit(X_train, Y_train)
```

```
▸ RFE
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

```
rfe.ranking_
```

```
array([61, 68, 78, 73, 66, 64, 62, 70, 72, 75, 69, 29, 53, 28, 30, 55, 81,
        1,  1,  1,  1,  1,  1,  1,  1,  1, 51, 74, 39, 67, 44, 50, 42, 54,
       26, 45,  1, 47, 43, 46, 58, 48, 33, 34, 32, 11, 10,  9, 16, 15, 17,
       25, 24, 40, 41, 49, 76, 14, 13, 12, 71, 20, 19, 18, 23, 22, 21, 57,
        7,  6,  5,  4,  3,  2, 27, 77,  8, 59, 82, 79, 31, 37, 38, 36, 80,
       35, 65, 63, 60, 52, 56])
```

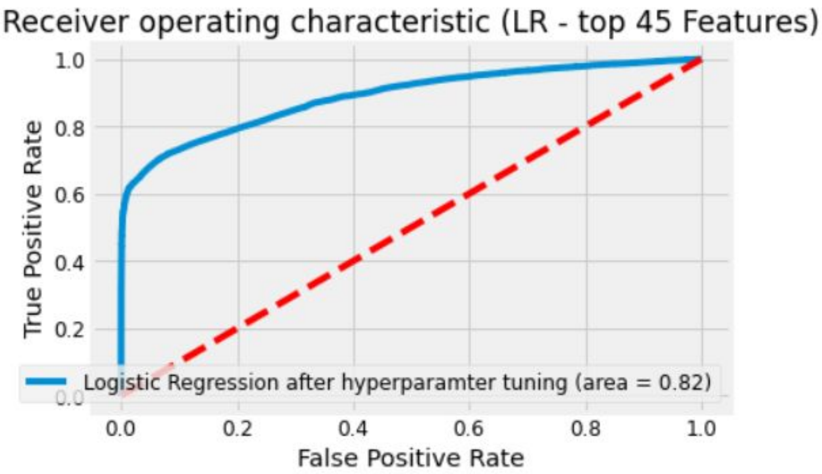
```
from operator import itemgetter
features = X_train.columns.to_l
for x, y in (sorted(zip(rfe.ran
    print(x, y)
```

```
1 age_[10-20)
1 age_[20-30)
1 age_[30-40)
1 age_[40-50)
1 age_[50-60)
1 age_[60-70)
1 age_[70-80)
1 age_[80-90)
1 age_[90-100)
1 medical_specialty_pediatrics
2 rosiglitazone_Up
3 rosiglitazone_Steady
4 rosiglitazone_No
5 pioglitazone_Up
6 pioglitazone_Steady
7 pioglitazone_No
8 miglitol_No
9 metformin_Up
10 metformin_Steady
11 metformin_No
12 glimepiride_Up
13 glimepiride_Steady
14 glimepiride_No
15 repaglinide_Steady
16 repaglinide_No
17 repaglinide_Up
18 glipizide_Up
19 glipizide_Steady
20 glipizide_No
21 glyburide_Up
```

LR model evaluation after Feature selection

Classification Report:

	precision	recall	f1-score	support
0	0.76	0.92	0.84	18697
1	0.90	0.71	0.80	18664
accuracy			0.82	37361
macro avg	0.83	0.82	0.82	37361
weighted avg	0.83	0.82	0.82	37361

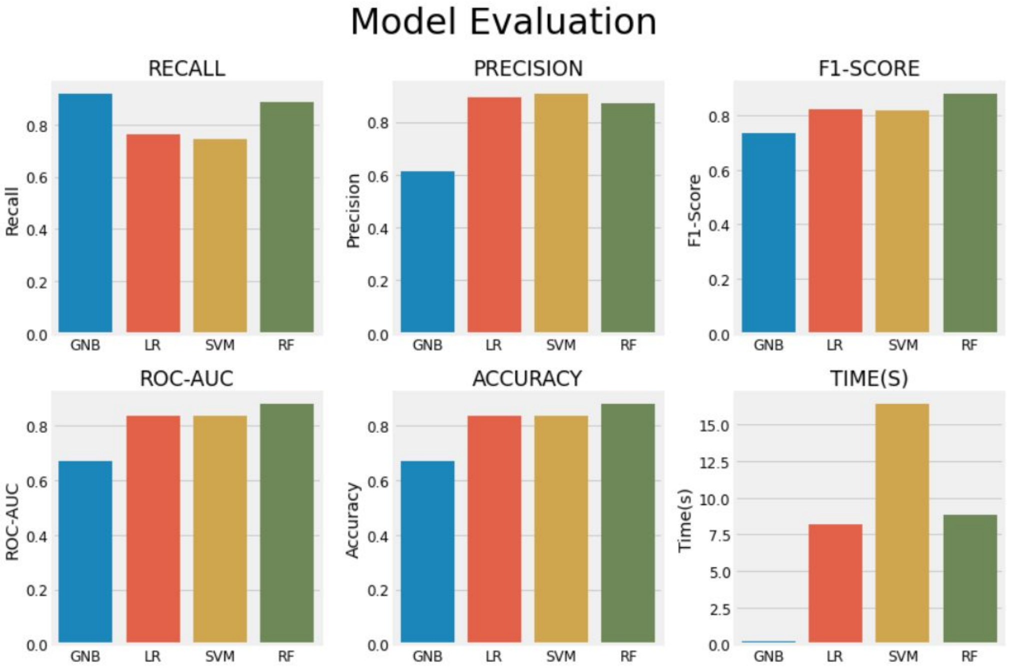



```
#Getting time difference before and after feature selection
print("Execution time of GridsearchCV (LR model) before Feature selection: ", round(total_time_before,3), "(s)")
print("Execution time of GridsearchCV (LR model) after Feature selection: ", round(total_time_after,3), "(s)")
```

```
Execution time of GridsearchCV (LR model) before Feature selection:  94.839 (s)
Execution time of GridsearchCV (LR model) after Feature selection:  66.496 (s)
```

Model Selection

- Out of all the linear models Logistic Regression has the best performance.
- If computational time is a critical factor then we could also go with Naives Bayes, however there's a slight tradedoff with performance.
- Regularized RF outperforms all other models but again that is to be expected since it's an ensemble model.



	Recall	Precision	F1-Score	ROC-AUC	Accuracy	Time(s)
Gaussian Naive Bayes	0.919096	0.613168	0.735592	0.670143	0.669923	0.202185
Logistic Regression	0.761412	0.894561	0.822634	0.835913	0.835979	8.193740
Support Vector Machine	0.745928	0.907562	0.818845	0.835043	0.835122	16.397628
Random Forest	0.887323	0.871769	0.879477	0.878517	0.878510	8.845504

Business Impact and Conclusion

- Hospital readmission is an important contributor to total medical expenditures and is an emerging indicator of quality of care.
- Diabetes, similar to other chronic medical conditions, is associated with increased risk of hospital readmission.
- hospital readmission is a high-priority health care quality measure and target for cost reduction, particularly within 30 days of discharge.
- The burden of diabetes among hospitalized patients is substantial, growing, and costly, and readmissions contribute a significant portion of this burden.
- Reducing readmission rates among patients with diabetes has the potential to greatly reduce health care costs while simultaneously improving care.
- Our aim is to provide some insights into the risk factors for readmission and also to identify the medicines that are the most effective in treating diabetes.

Business Impact and Conclusion

- Although most of the identified risk factors such as being female, being aged ≥65 years, and having comorbidities like diabetes are not modifiable, an understanding of their impact on disease outcomes is relevant to health professionals and policymakers for developing and updating clinical practice guidelines to reduce 30-day unplanned hospital readmission.
- Better management and monitoring of multiple comorbidities associated with diabetes is recommended to delay the progression of complications associated with DM, thus reducing the risk of 30-day unplanned hospital readmission
- Through this project, we created a machine learning model that is able to predict the patients with diabetes with highest risk of being readmitted within 30 days.
- The best linear model was a Logistic Regression with optimized hyperparameters. & Regularized RF is best overall model


شكراً جزيلاً
рахмат
danke
謝謝
ngiyabonga
Баярлалаа
spas
teşekkür ederim
mersi
kia ora
barka
welalin
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
kiitos
dankie
nanni
nandri
dhanyavad
thank you
gracias
хвала
asante
manana
obrigada
tenki
bedankt
bayarlalaa
gràcie
hvala
maururu
köszönöm
aciū
dankon
akun
djiere dieuf
tau
mochchakkeram
murakoze
enkosi
dziękuję
дякую
mamnun
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
chokrane
sobodi
dėkuji
sukriya
kop khun krap
taiku
obrigado
mèsi
sagolun
ありがとう
arigatô
takk
dakujem
trugarez
tanemirt
grazie
didi madloba
najis tuke
terima kasih
rahmet
diolch
dhanyavadagalu
shukriya
mercê
мерси
kam sah hamnida
rahmat
তোমাকে ধন্যবাদ
감사합니다
xiexie
ευχαριστώ
merci