IDENTIFY DIABETIC PATIENT READMISSION

GROUP – 7:

ABHINAV SALUJA

BHAVIKA FALDU

CHINMAY PARKAR

MANISH SHUKLA

INTRODUCTION

OBJECTIVE:

To predict the hospital re-admission probability of a DIABETIC patient by using appropriate Data Science techniques.

- A clinic readmission is the point at which a patient who is released from the emergency clinic, gets re-conceded again inside a specific timeframe.
- Clinic readmission rates for specific conditions are currently viewed as a marker of medical clinic quality and furthermore influence the expense of care antagonistically.
- Thus, Centers for Medicare and Medicaid Services set up the Hospital Readmissions Reduction Program which means to improve nature of care for patients and decrease social insurance spending by applying installment punishments to emergency clinics that have more than anticipated readmission rates for specific conditions.
- In 2011, American medical clinics spent over $41 billion on diabetic patients who got readmitted inside 30 days of release.
- Having the option to decide factors that lead to higher readmission in such patients, and correspondingly having the option to anticipate which patients will get readmitted can assist medical clinics with sparing a great many dollars while improving nature of care.

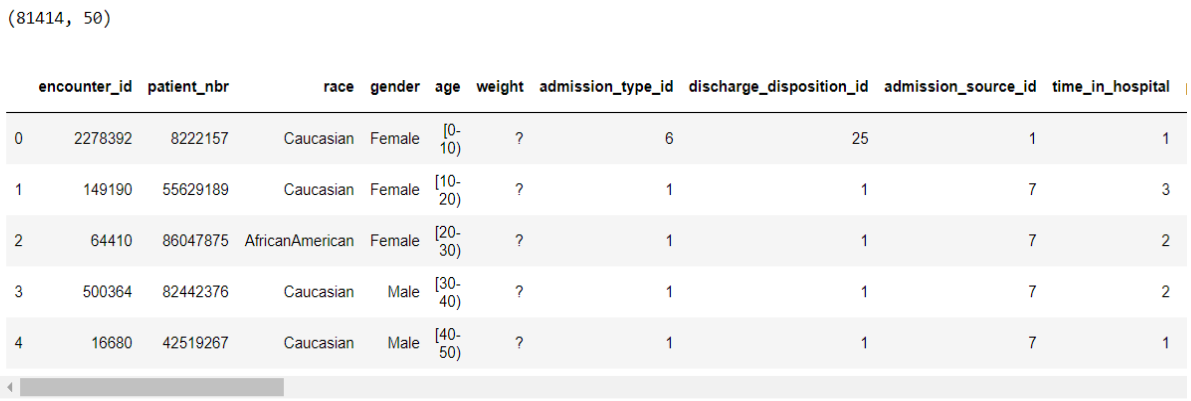
Below are questions need to be answered from the analysis:

- What variables are the most grounded indicators of emergency clinic readmission in diabetic patients?
- How well would we be able to foresee medical clinic readmission in this dataset with constrained highlights?

About DATA:

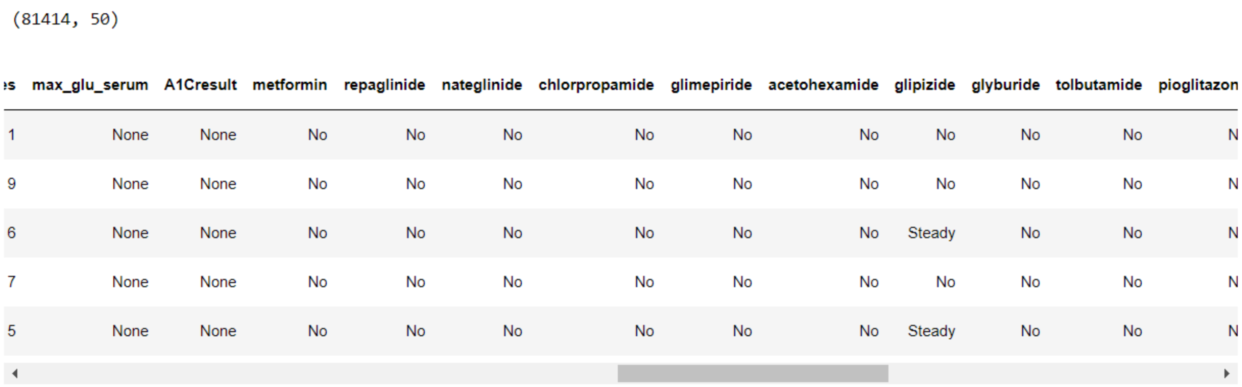
Data is retrieved from UCI (University of California, Irvine) repository

- Encountered data are collected from 130 hospitals on Diabetic patient for a period of 1999-2008
- Dataset has over 50 features, including patient characteristics, conditions, tests and 23 medications.



```
(81414, 50)
```

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	1	1
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	3
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	2
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	2
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	1



```
(81414, 50)
```

es	max_glu_serum	A1Cresult	metformin	repaglinide	nateglinide	chlorpropamide	glimepiride	acetohexamide	glipizide	glyburide	tolbutamide	pioglitazon
1	None	None	No	No	No	No	No	No	No	No	No	N
9	None	None	No	No	No	No	No	No	No	No	No	N
6	None	None	No	No	No	No	No	No	Steady	No	No	N
7	None	None	No	No	No	No	No	No	No	No	No	N
5	None	None	No	No	No	No	No	No	Steady	No	No	N

Why this DATA?

- People Affected by Diabetes: WORLD: 425M, USA: 26M (8.3% of the population)
- Expenditure on Diabetes: WORLD: 727B Dollars, USA: 327B Dollars
- People that will be affected by Diabetes: 629 million
- Penalties paid by US Hospitals due to readmission of patients: 528 million Dollars
- Readmission Rate of diabetes patients that are readmitted within 30-days of discharge: 20.3%

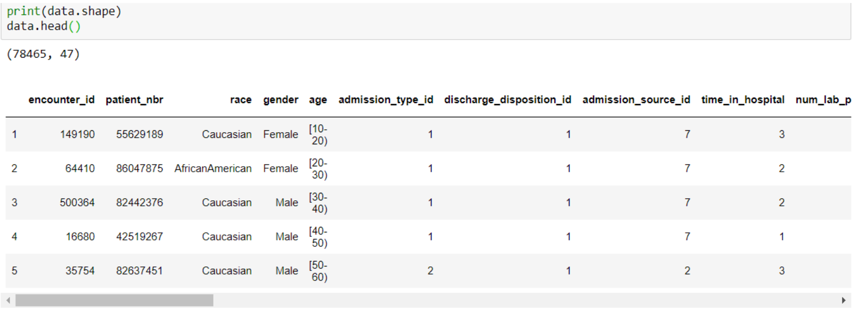
STEP: 1 - Data Cleaning

It is always advisable to have clean and processed data for modelling purposes. This has been performed using the following steps:

1. Carrying out null value treatment of the data.
2. Performing EDA to find the highly correlated variables, analyze the trend and distribution of the data.
3. Sub-diving the columns based on the EDA.
4. Creating separate test and train dataset.
5. Carrying out scaling of the data (if required).

Below are the steps carried out in cleaning the data:

- Loading the required libraries for downstream activities.
- Reading the data from the csv files, which will be used for building the model.
- Checking the null values for data loaded (But no null values found).
- Displaying the unique values and percentage of unique values for each feature in order to identify other type of missing values and variation explained by each variable.
- Analyzing the features - weight, payer_code and medical_speciality, we notice that these features have approximately 40% or more than 40% of data missing and represented by '?'.
- In variable 'Gender', missing value is shown by 'Unknown/Invalid', which are approximately 5 records.
- Roughly around more than 40% of data is missing from the features 'weights', 'payer_code' and 'medical_speciality', hence these columns have been dropped.
- Deleting the entire record for column 'Gender', which have values as 'Unknown/Invalid'.
- Below is the image displaying the look of the data after cleaning



```
print(data.shape)
data.head()
```

```
(78465, 47)
```

	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_p
1	149190	55629189	Caucasian	Female	[10-20)	1	1	7	3	
2	64410	86047875	AfricanAmerican	Female	[20-30)	1	1	7	2	
3	500364	82442376	Caucasian	Male	[30-40)	1	1	7	2	
4	16680	42519267	Caucasian	Male	[40-50)	1	1	7	1	
5	35754	82637451	Caucasian	Male	[50-60)	2	1	2	3	

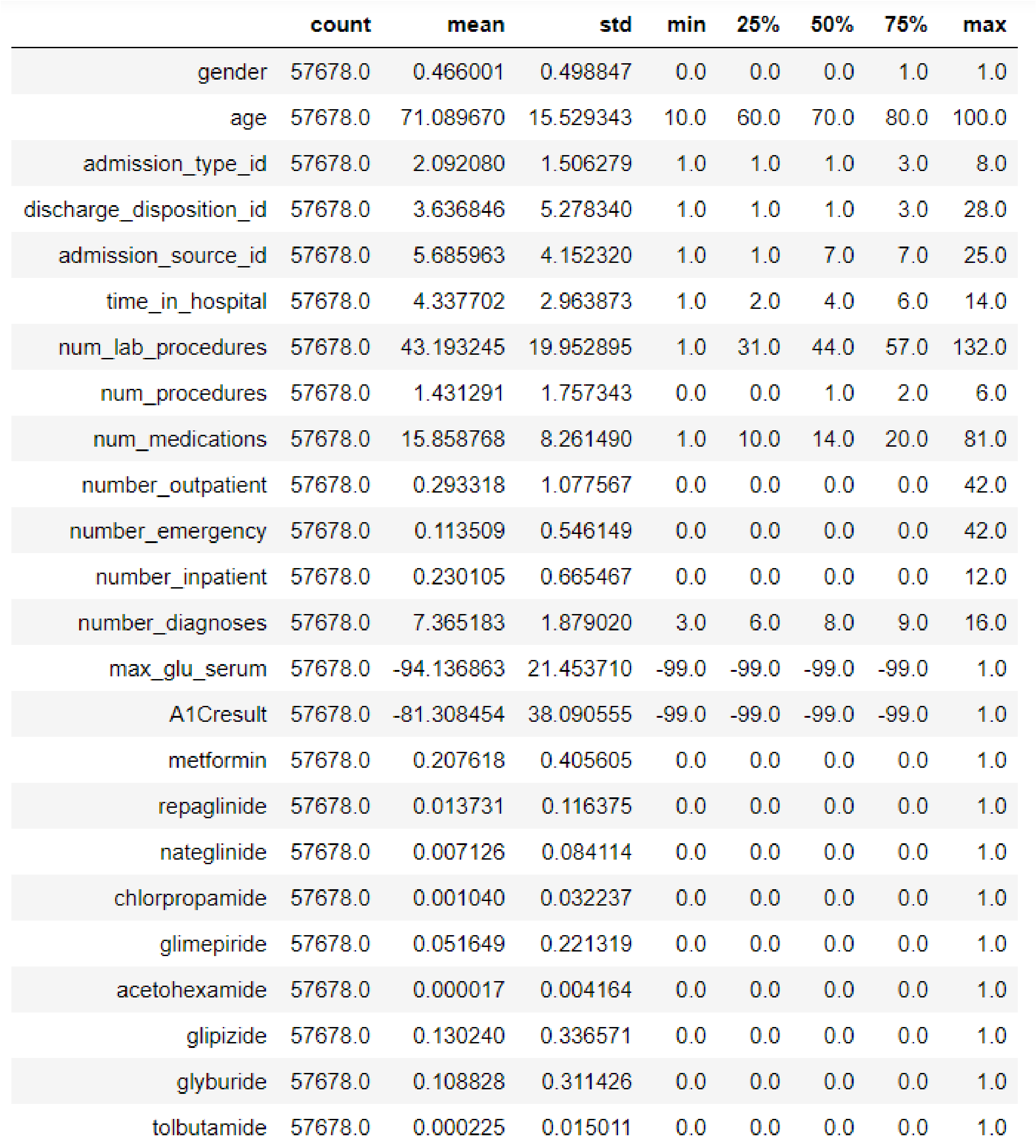
STEP: 2 - Feature Engineering and Creation

We would also like to spend some time on the feature engineering. There are lot of categorical features and to input those into model by creating dummies will not only be computationally inefficient but will also reduce the predictive power of the model because of too many indicator variables.

Below are the steps carried out in feature engineering and feature creation phase:

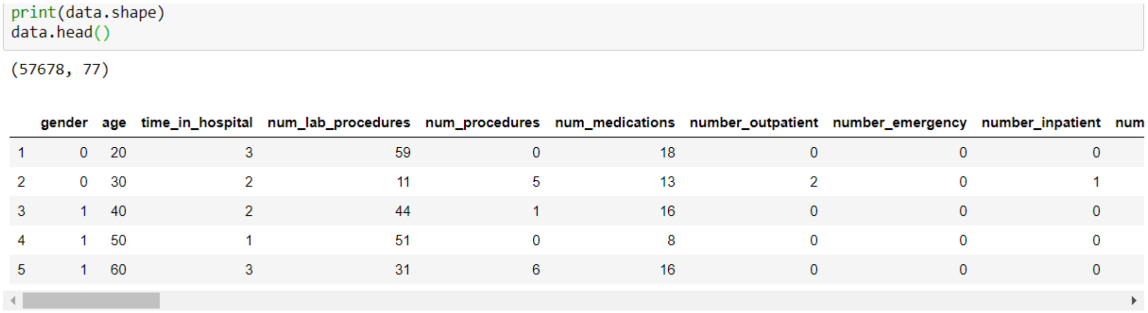
- **Number of medicines utilized**: Total number of prescriptions utilized by the patient, so we made component by checking the total number medications utilized during the experience.
- **Number of prescription changes**: Data contains 23 variables for 23 medicines, regardless of whether an adjustment in that drug was made or not, so we chose to tally what number of changes were made altogether for every patient.
- **Service usage**: Added information contains factors for number of inpatient visits, emergency room visits and outpatient visits
- Encoded the 'medication change' feature from 'No' (no change) and 'Ch' (changed) into 0 and 1
- Similarly, converted 'Gender' variable from 'Male' and 'Female' into 1 and 0
- Also, changed 'diabetesMed' (Diabetic Medicine use) column from 'Yes' and 'No' into 1 and 0
- Dataset has multiple occurrence of same patient, which brings in biasedness. So, keeping the first instance of the patient encounter and removing the other instances, with the help of column patient_nbr.
- 'Diag_1','Diag_2' and 'Diag_3' have almost 700-900 categories, including these variables with 700-900 dummies, will definitely make our model complex and time consuming.
- 'Encounter_id' and 'patient_nbr' are columns, which just contain Id to uniquely identify each record that would not help explain much variance about the data.
- Based on above insights, dropping all the 5 columns, to remove redundant part from the data.
- Converting remaining categorical variables into numerical and features such as 'readmitted', 'A1Cresult' and 'max_glu_serum into binary values 1 and 0.

- 'Age' feature has 10 categories, assuming maximum values of each categories. For Ex, Age as 10 for category value '0-10'
- Generating the basic statistics such as mean, standard deviation, min, max, etc. for the continuous variable
- Observed that 4 medicines out of 23 have mean value less than 0.1, which signifies dominance of one value in the feature.
- Below is the image, showing basic descriptive statistics of the feature:



	count	mean	std	min	25%	50%	75%	max
gender	57678.0	0.466001	0.498847	0.0	0.0	0.0	1.0	1.0
age	57678.0	71.089670	15.529343	10.0	60.0	70.0	80.0	100.0
admission_type_id	57678.0	2.092080	1.506279	1.0	1.0	1.0	3.0	8.0
discharge_disposition_id	57678.0	3.636846	5.278340	1.0	1.0	1.0	3.0	28.0
admission_source_id	57678.0	5.685963	4.152320	1.0	1.0	7.0	7.0	25.0
time_in_hospital	57678.0	4.337702	2.963873	1.0	2.0	4.0	6.0	14.0
num_lab_procedures	57678.0	43.193245	19.952895	1.0	31.0	44.0	57.0	132.0
num_procedures	57678.0	1.431291	1.757343	0.0	0.0	1.0	2.0	6.0
num_medications	57678.0	15.858768	8.261490	1.0	10.0	14.0	20.0	81.0
number_outpatient	57678.0	0.293318	1.077567	0.0	0.0	0.0	0.0	42.0
number_emergency	57678.0	0.113509	0.546149	0.0	0.0	0.0	0.0	42.0
number_inpatient	57678.0	0.230105	0.665467	0.0	0.0	0.0	0.0	12.0
number_diagnoses	57678.0	7.365183	1.879020	3.0	6.0	8.0	9.0	16.0
max_glu_serum	57678.0	-94.136863	21.453710	-99.0	-99.0	-99.0	-99.0	1.0
A1Cresult	57678.0	-81.308454	38.090555	-99.0	-99.0	-99.0	-99.0	1.0
metformin	57678.0	0.207618	0.405605	0.0	0.0	0.0	0.0	1.0
repaglinide	57678.0	0.013731	0.116375	0.0	0.0	0.0	0.0	1.0
nateglinide	57678.0	0.007126	0.084114	0.0	0.0	0.0	0.0	1.0
chlorpropamide	57678.0	0.001040	0.032237	0.0	0.0	0.0	0.0	1.0
glimepiride	57678.0	0.051649	0.221319	0.0	0.0	0.0	0.0	1.0
acetohexamide	57678.0	0.000017	0.004164	0.0	0.0	0.0	0.0	1.0
glipizide	57678.0	0.130240	0.336571	0.0	0.0	0.0	0.0	1.0
glyburide	57678.0	0.108828	0.311426	0.0	0.0	0.0	0.0	1.0
tolbutamide	57678.0	0.000225	0.015011	0.0	0.0	0.0	0.0	1.0

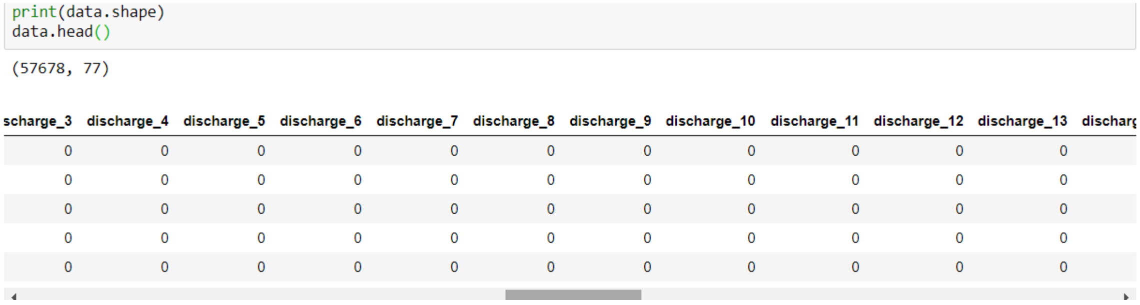
- Dropping the 19 medicine columns, since they do not explain much variance and hence are not adding any meaning to our model
- Checking the datatype of each variable in the dataset, almost all variables are integer type.
- Creating dummies for the column race, admission_type, discharge_disposition and admission_source
- Variable race will have 5 dummies, admission_type 8 dummies, discharge_disposition approx. 26 dummies and admission_source approximately 20 dummies
- Deleting the main column from the dataset, after concatenating dummies for all the above columns in the main dataset.
- Below is the image showing the shape and look of dataset after feature engineering and feature creation:



```
print(data.shape)
data.head()
```

```
(57678, 77)
```

	gender	age	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_outpatient	number_emergency	number_inpatient	num
1	0	20	3	59	0	18	0	0	0	
2	0	30	2	11	5	13	2	0	1	
3	1	40	2	44	1	16	0	0	0	
4	1	50	1	51	0	8	0	0	0	
5	1	60	3	31	6	16	0	0	0	



```
print(data.shape)
data.head()
```

```
(57678, 77)
```

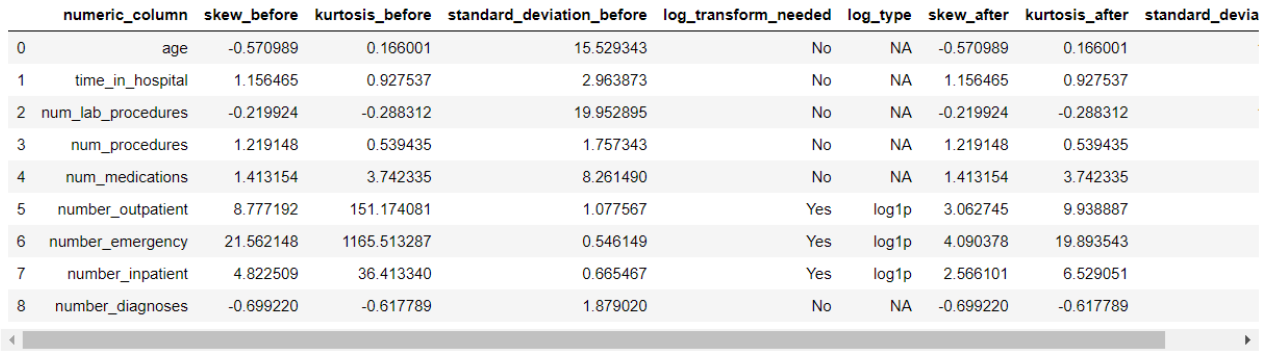
scharge_3	discharge_4	discharge_5	discharge_6	discharge_7	discharge_8	discharge_9	discharge_10	discharge_11	discharge_12	discharge_13	discharg
0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	

STEP: 3 - Transformation and Outlier Removal

As a reference, the skew of a normal distribution is 0 and the excess kurtosis (difference of actual kurtosis from ideal normal distribution value of 3), as returned by the kurtosis() function for a normal distribution is 0, which would impact scaling.

Below are the steps involved in Log Transformation and Outlier Removal:

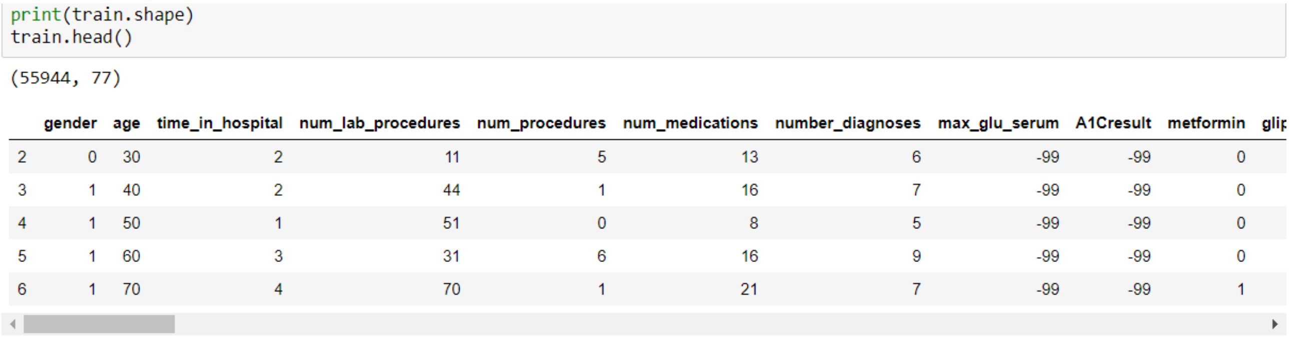
- Checking if features have skewness and have high kurtosis, which would impact standardization.
- Three columns need to be transformed, performing log transformation where a skew or kurtosis beyond the limits of -2 ≤ skew and kurtosis ≤ 2
- Image below shows the skewness and kurtosis analysis for continuous variables:



	numeric_column	skew_before	kurtosis_before	standard_deviation_before	log_transform_needed	log_type	skew_after	kurtosis_after	standard_devia
0	age	-0.570989	0.166001	15.529343	No	NA	-0.570989	0.166001	
1	time_in_hospital	1.156465	0.927537	2.963873	No	NA	1.156465	0.927537	
2	num_lab_procedures	-0.219924	-0.288312	19.952895	No	NA	-0.219924	-0.288312	
3	num_procedures	1.219148	0.539435	1.757343	No	NA	1.219148	0.539435	
4	num_medications	1.413154	3.742335	8.261490	No	NA	1.413154	3.742335	
5	number_outpatient	8.777192	151.174081	1.077567	Yes	log1p	3.062745	9.938887	
6	number_emergency	21.562148	1165.513287	0.546149	Yes	log1p	4.090378	19.893543	
7	number_inpatient	4.822509	36.413340	0.665467	Yes	log1p	2.566101	6.529051	
8	number_diagnoses	-0.699220	-0.617789	1.879020	No	NA	-0.699220	-0.617789	

- Computing log(x) for any feature x if percentage of 0s in x ≤ 2%, after removing the zeros, which ensures that we don't bulk-remove records that hold predictive power for other columns
- Compute log1p(x) otherwise (log1p(x) means log(x+1), while retaining the zeros
- Anything within 3 Standard Deviations on either side of the mean would include 99.7% of our data and the remaining 0.3% we treat as outliers
- Using this logic, we restricted the data to within 3 Standard deviations on either side from the mean for each numeric column.

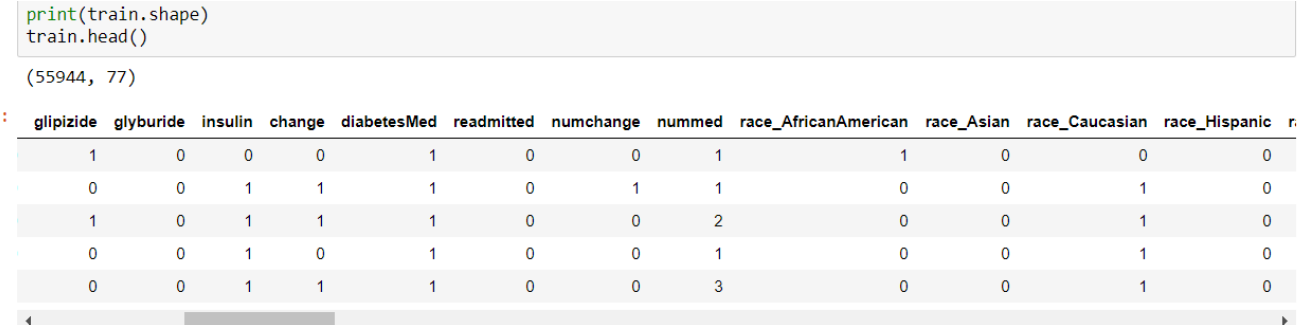
- Below are the images, which show data after log transformation and outlier removal:



```
print(train.shape)
train.head()
```

```
(55944, 77)
```

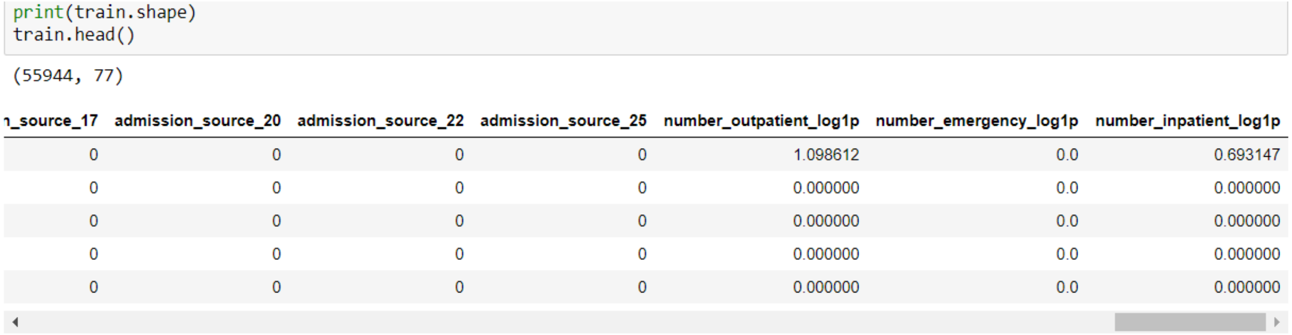
	gender	age	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_diagnoses	max_glu_serum	A1Cresult	metformin	glip
2	0	30	2	11	5	13	6	-99	-99	0	
3	1	40	2	44	1	16	7	-99	-99	0	
4	1	50	1	51	0	8	5	-99	-99	0	
5	1	60	3	31	6	16	9	-99	-99	0	
6	1	70	4	70	1	21	7	-99	-99	1	



```
print(train.shape)
train.head()
```

```
(55944, 77)
```

glipizide	glyburide	insulin	change	diabetesMed	readmitted	numchange	nummed	race_AfricanAmerican	race_Asian	race_Caucasian	race_Hispanic	r
1	0	0	0	1	0	0	1	1	0	0	0	
0	0	1	1	1	0	1	1	0	0	1	0	
1	0	1	1	1	0	0	2	0	0	1	0	
0	0	1	0	1	0	0	1	0	0	1	0	
0	0	1	1	1	0	0	3	0	0	1	0	



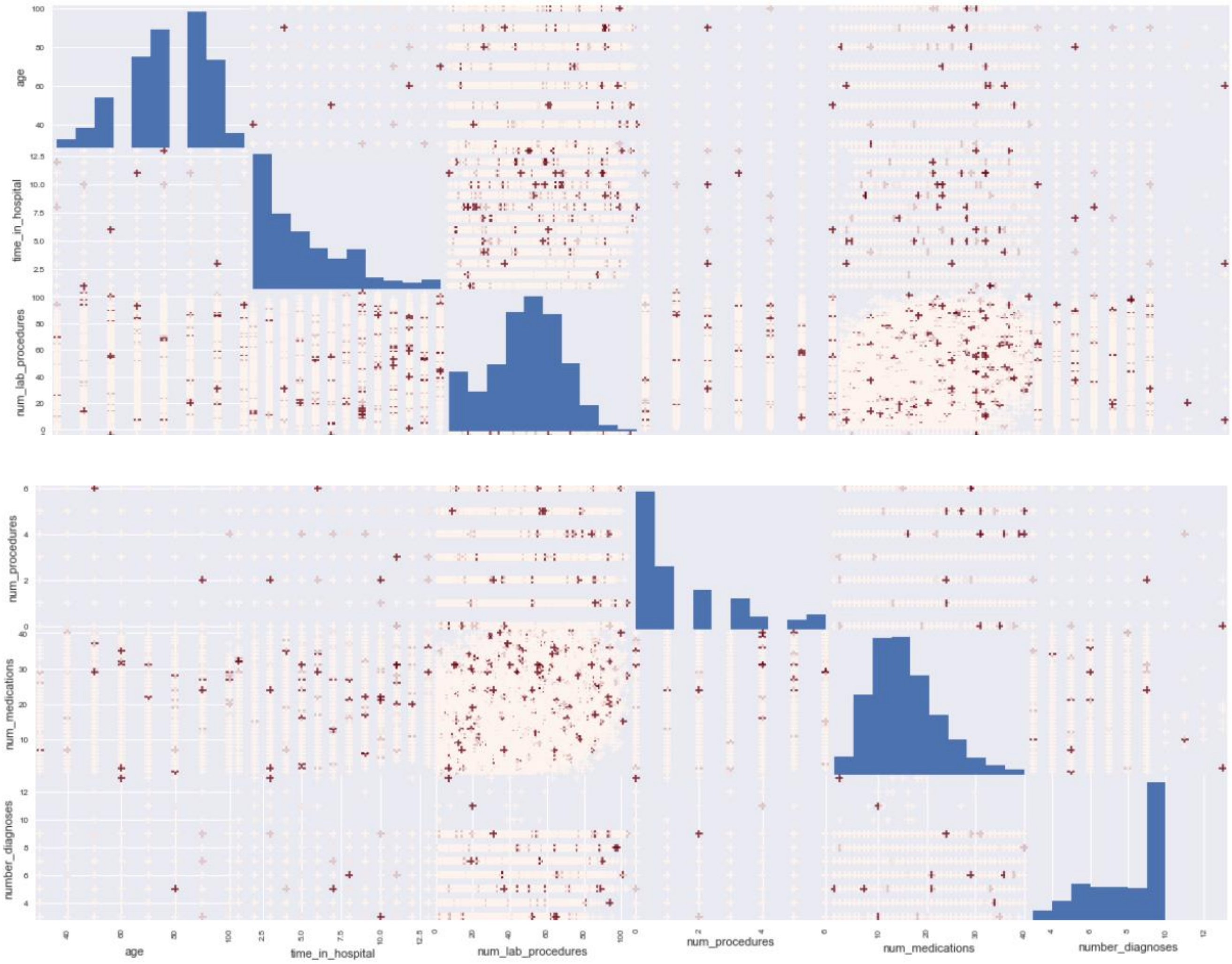
```
print(train.shape)
train.head()
```

```
(55944, 77)
```

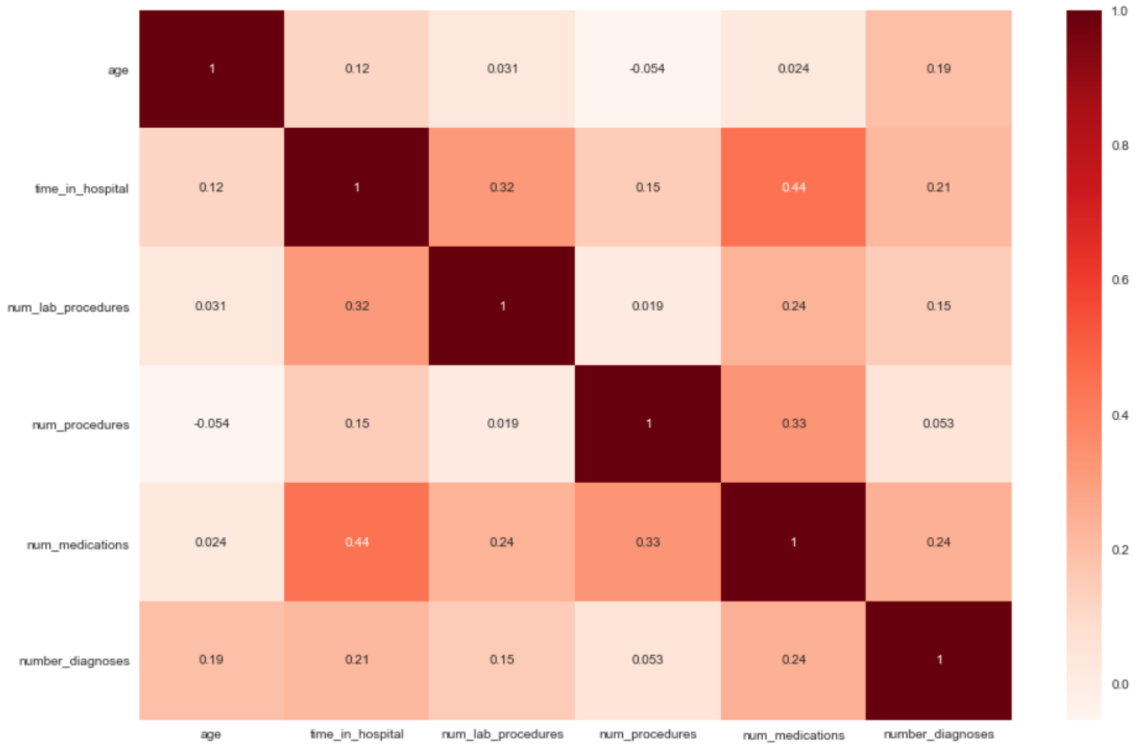
ı_source_17	admission_source_20	admission_source_22	admission_source_25	number_outpatient_log1p	number_emergency_log1p	number_inpatient_log1p
0	0	0	0	1.098612	0.0	0.693147
0	0	0	0	0.000000	0.0	0.000000
0	0	0	0	0.000000	0.0	0.000000
0	0	0	0	0.000000	0.0	0.000000
0	0	0	0	0.000000	0.0	0.000000

STEP: 4 - Exploratory Data Analysis and Sampling

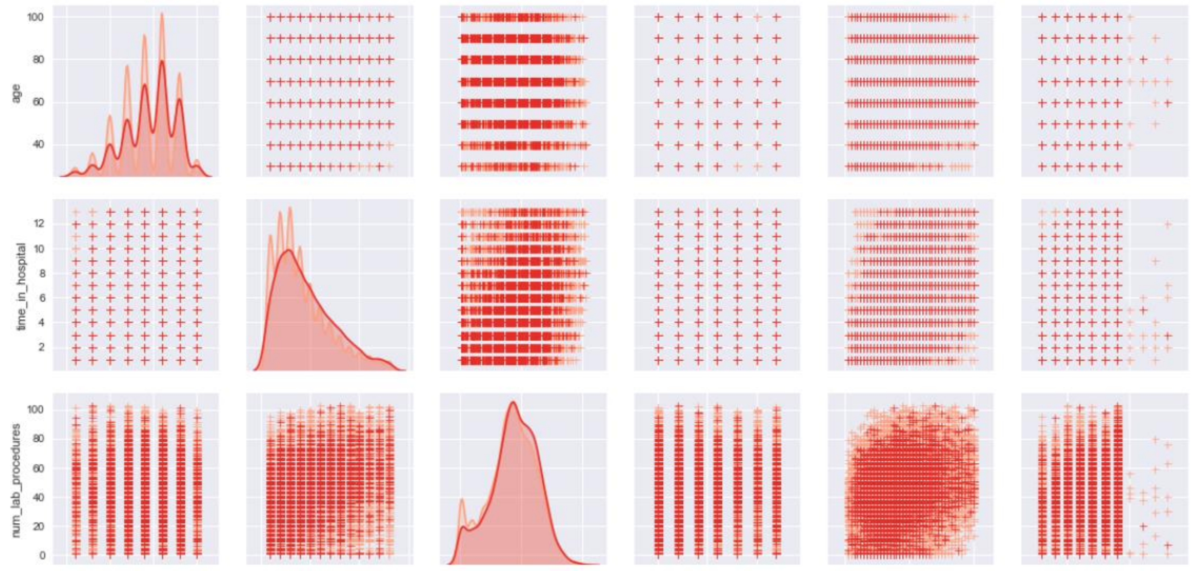
- Plotting scatter matrix to check whether variables are related and whether variables relation is positive or negative.

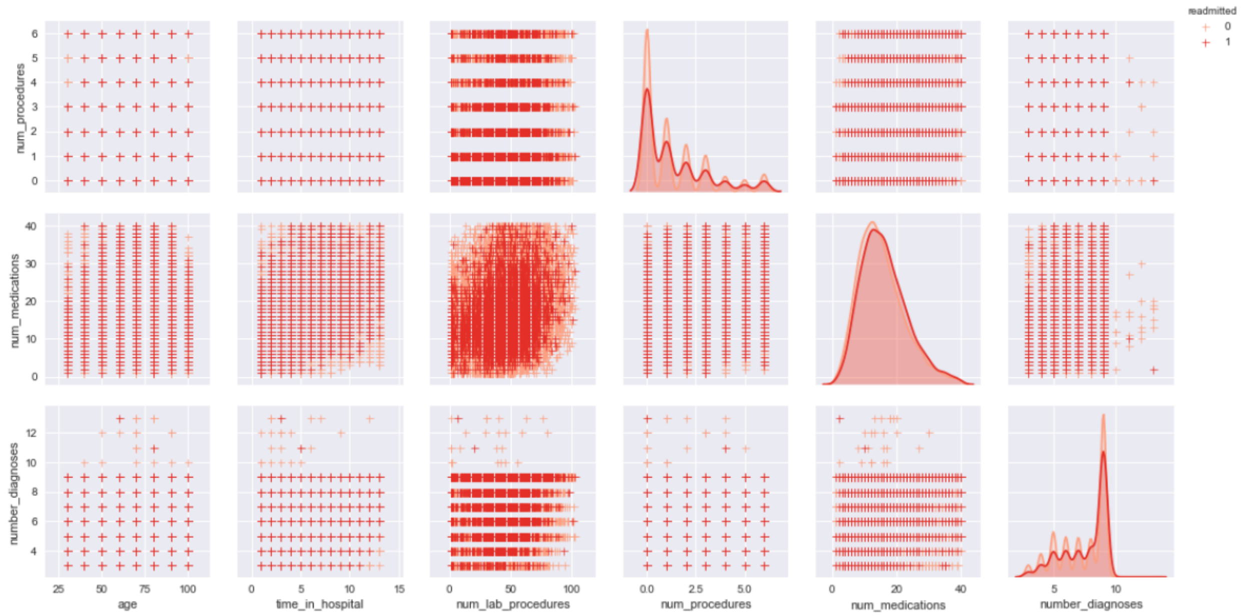


- Graphing Heatmap to check whether interaction terms of variables have any positive or negative correlation

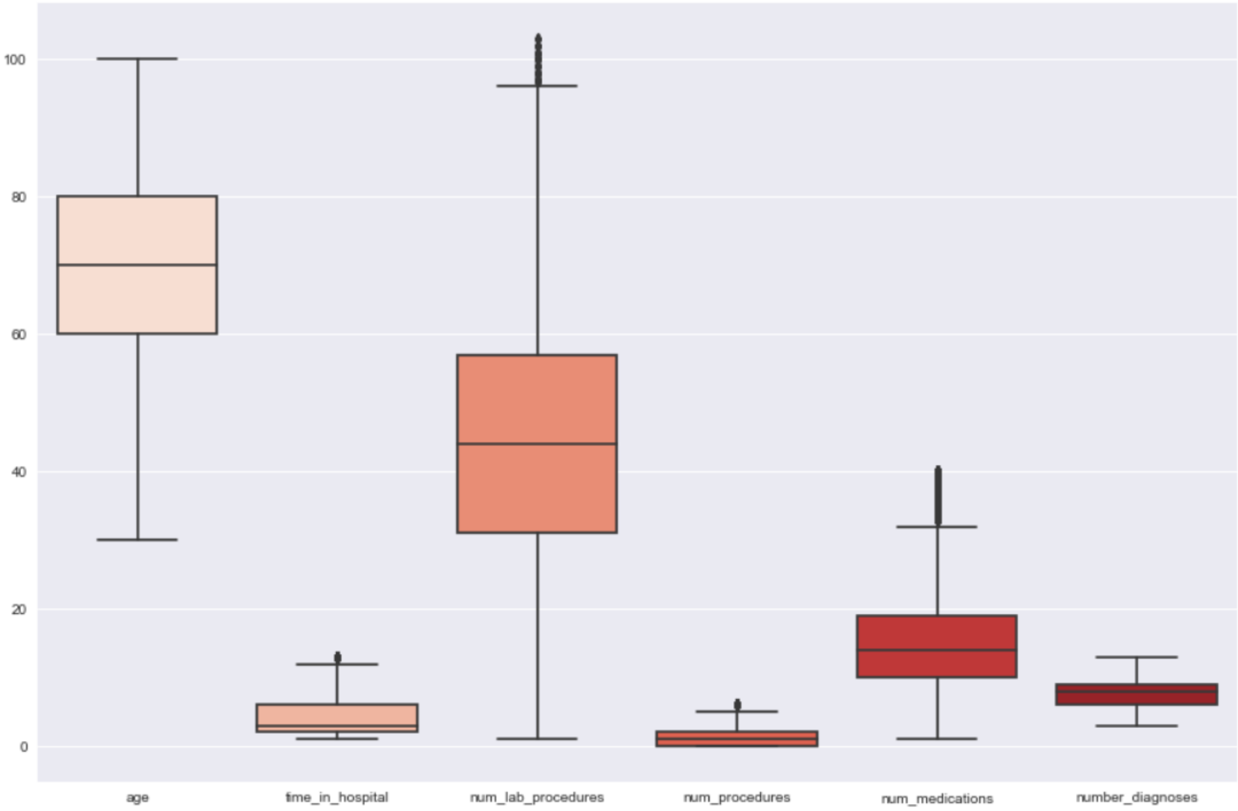


- Displaying pair-plot to have histogram and scatterplot together, in order to find the distribution of the variables and see relationship between the variables

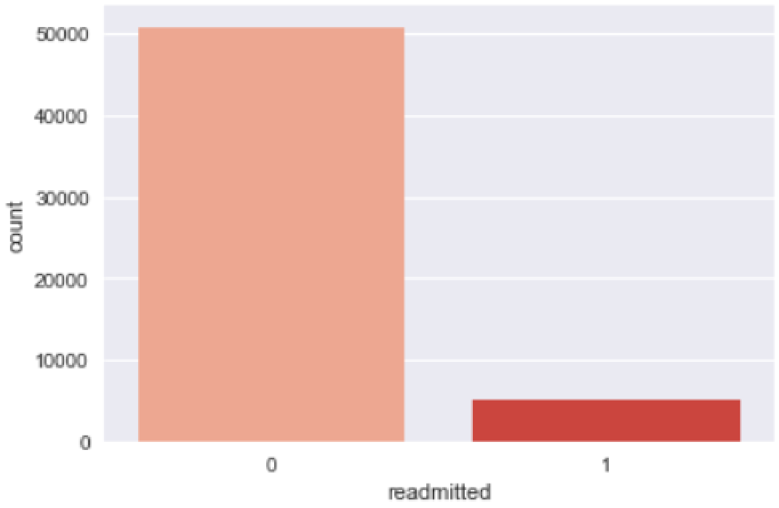




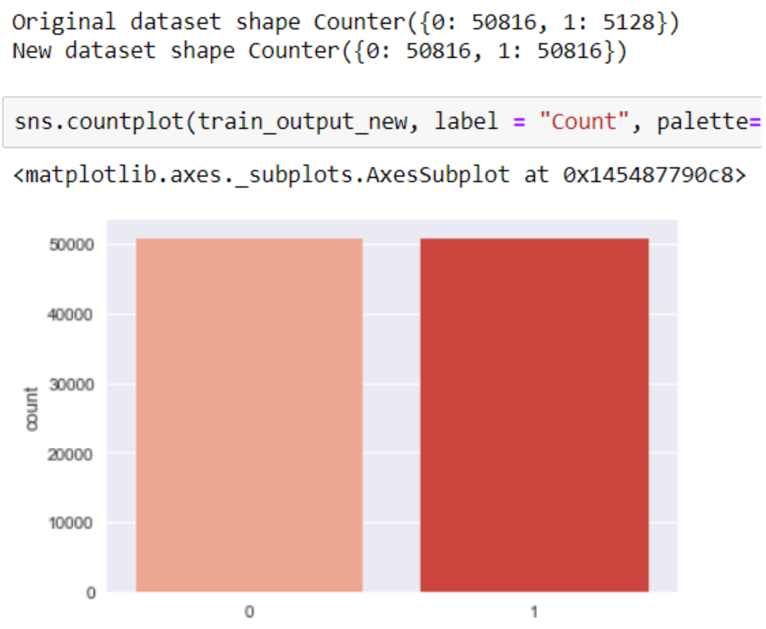
- Labelling boxplot to see the amount of variation explained by continuous variables such as age, time in hospital, number of lab procedures, etc.



- Checking the counts for number of 1's and 0's in target variable 'readmitted'
- By Analyzing, found that number of 0's are more and dataset is highly imbalanced, which will lead the model to predict 0's correctly but not 1's



- Separating the dataset from dependent variable 'readmitted' and all other independent variables
- Since the dataset is highly imbalanced, applying oversampling by SMOTE to make equal number of 0's and 1's
- Below, is the image showing the shape of dataset after oversampling technique.



```
Original dataset shape Counter({0: 50816, 1: 5128})
New dataset shape Counter({0: 50816, 1: 50816})
```

```
sns.countplot(train_output_new, label = "Count", palette=
```

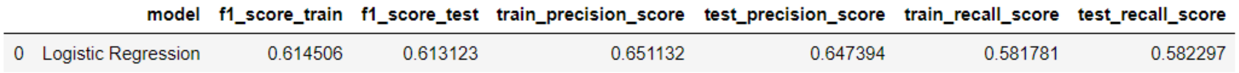
```
<matplotlib.axes._subplots.AxesSubplot at 0x145487790c8>
```

STEP: 5 - Model Building and Evaluation

- Splitting the dataset into train and test, 80% of data will be used in training the model and 20% of data in evaluating the model
- Once the data is split, applying Min-Max Scaling to bring all the variable on one scale and not having influence of variables magnitude on model

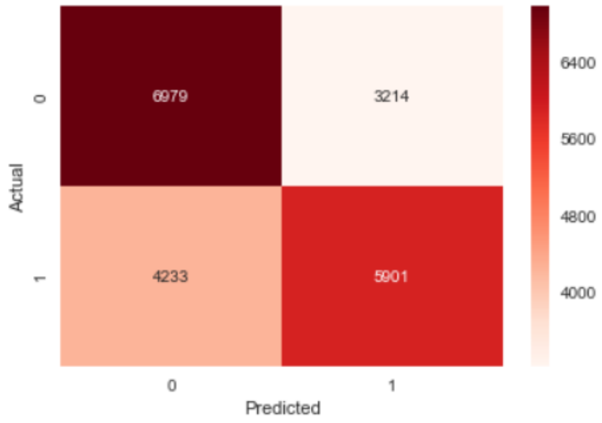
1) Logistic Regression:

- Using Logistic Regression to see the relative impact of each variable and statistical significance of each factor on the probability of readmission.
- Training the Logistic Regression model on hyperparameter 'penalty' using Grid search and cross validation of 5.
- By Grid Search, we found that 'penalty'('l1') gives us the best result, using the best parameters to train our model.
- Once we trained our model on best parameters, calculated evaluation parameter such as recall, precision and f1 score to judge the model
- We found that, model has low evaluation parameters and concluded to use some other algorithm to build our model.



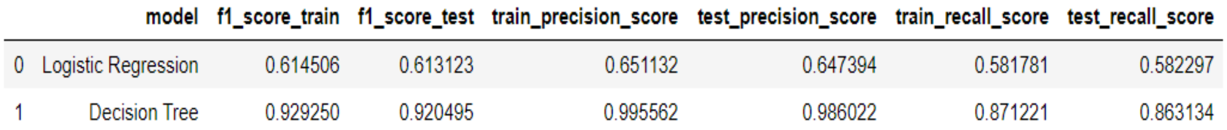
	model	f1_score_train	f1_score_test	train_precision_score	test_precision_score	train_recall_score	test_recall_score
0	Logistic Regression	0.614506	0.613123	0.651132	0.647394	0.581781	0.582297

- Plotting confusion matrix for Logistic Regression, to see the number True positive, True Negative, False Positive and False Negative



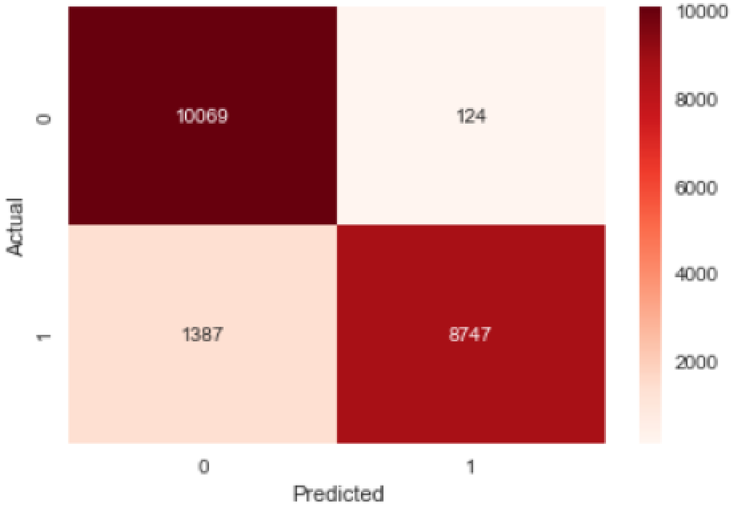
2) Decision Tree:

- Using Decision Tree to capture the nonlinear effects of each features and interaction between variables.
- Training the Decision Tree model on hyperparameter 'max_depth' using Grid search and cross validation of 5
- By Grid Search, we found that 'max_depth = 12' gives us the best result, using the best parameters to train our model
- Calculating the evaluation metrics, after training the model, in order to judge it
- We found that built Decision tree model has overall good evaluation metrics



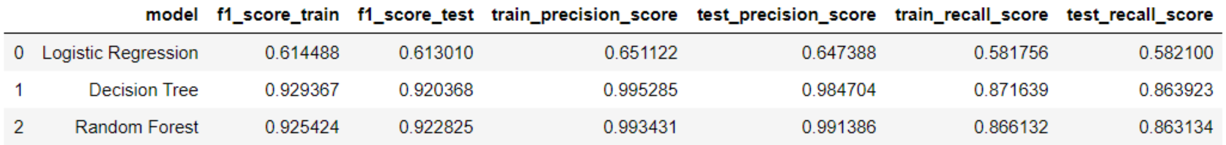
	model	f1_score_train	f1_score_test	train_precision_score	test_precision_score	train_recall_score	test_recall_score
0	Logistic Regression	0.614506	0.613123	0.651132	0.647394	0.581781	0.582297
1	Decision Tree	0.929250	0.920495	0.995562	0.986022	0.871221	0.863134

- Displaying confusion parameters to find number of correct and wrong prediction by the model for readmission or not



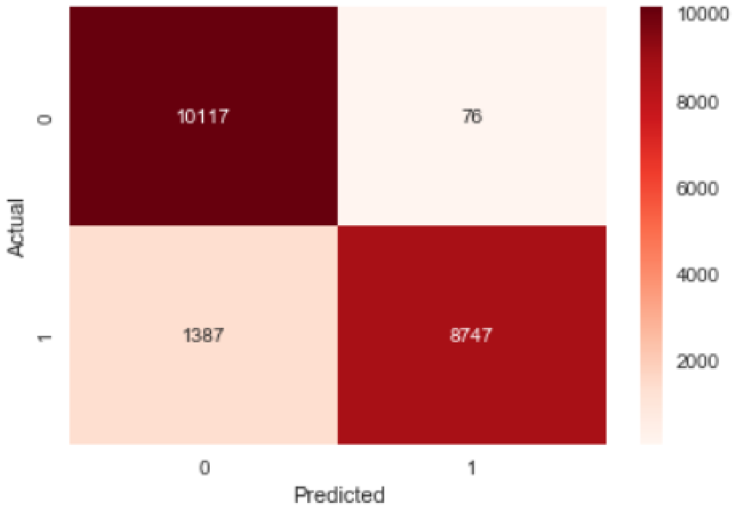
3) Random Forest:

- Since single tree has improved evaluation metrics so much, using many Decision trees with randomly assigned subsets of features, which is called Random Forest
- Training the Random Forest model on hyperparameter 'max_depth', 'max_features', 'min_samples_split', 'min_samples_leaf' and 'Bootstrap', using Random search and iteration of 20
- By Random Search and multiple iterations, we found that bootstrap=False, max_depth=8, max_features=22, min_samples_leaf=1, min_samples_split=5 gives us the best result, using the best parameters to train our model
- Once we trained our model on best parameters, evaluation parameter such as recall, precision and f1 score was calculated to judge the model
- We found that, model has best evaluation parameters in comparison to other algorithms and concluded Random Forest is the ‘Go To’ algorithm for model deployment.



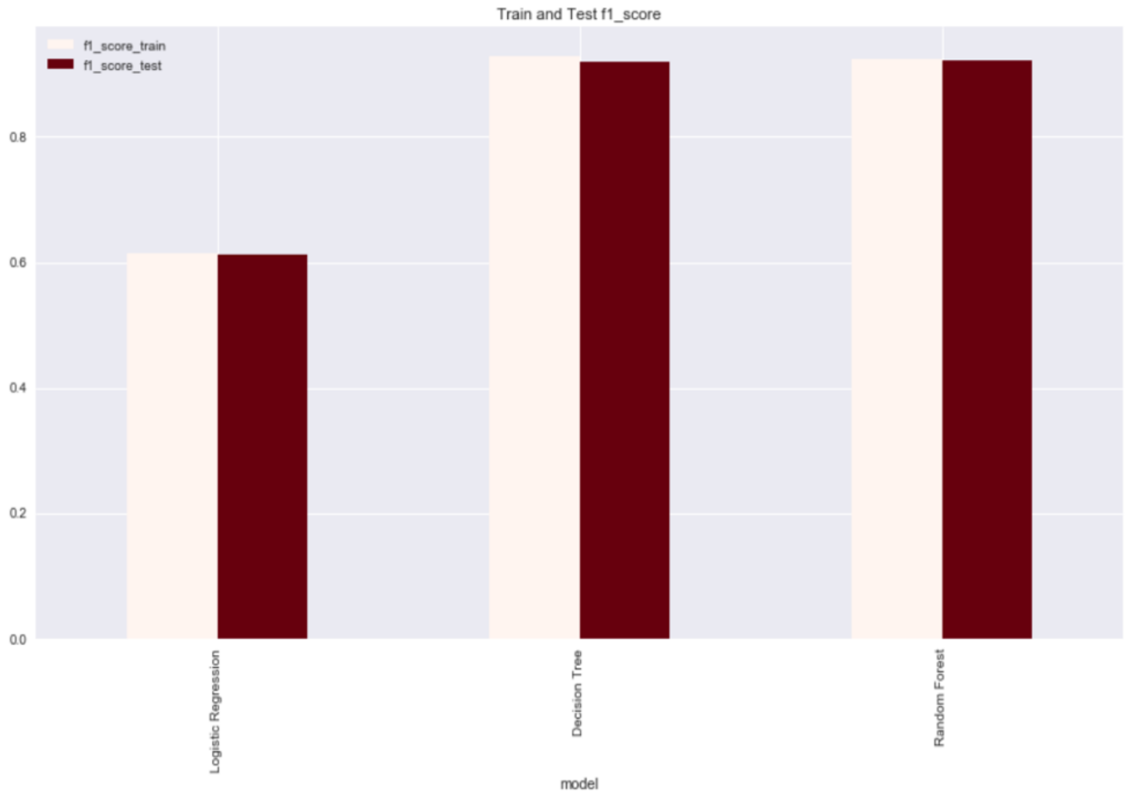
	model	f1_score_train	f1_score_test	train_precision_score	test_precision_score	train_recall_score	test_recall_score
0	Logistic Regression	0.614488	0.613010	0.651122	0.647388	0.581756	0.582100
1	Decision Tree	0.929367	0.920368	0.995285	0.984704	0.871639	0.863923
2	Random Forest	0.925424	0.922825	0.993431	0.991386	0.866132	0.863134

- Labelling the actual and predicted numbers of 1's and 0's by Random Forest model

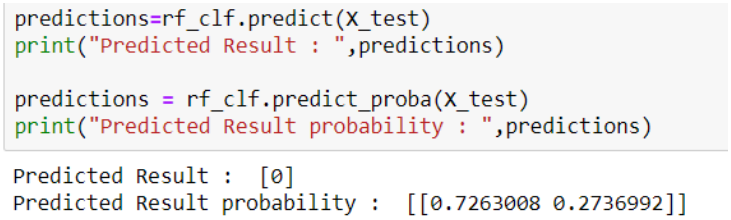


STEP: 6 - Best Model and Deployment

- Comparing 3 algorithms model on f1_score evaluation metric
- We found that Random Forest gives us the best result, on the basis of train and test f1_score
-



- Deploying the random forest model and min-max scaler transformation to create the pickle files and dump it
- Once the pickle files are created, passed one data to get the prediction and find prediction probability
- First prediction probability says the probability of not readmitting and second one of readmitting

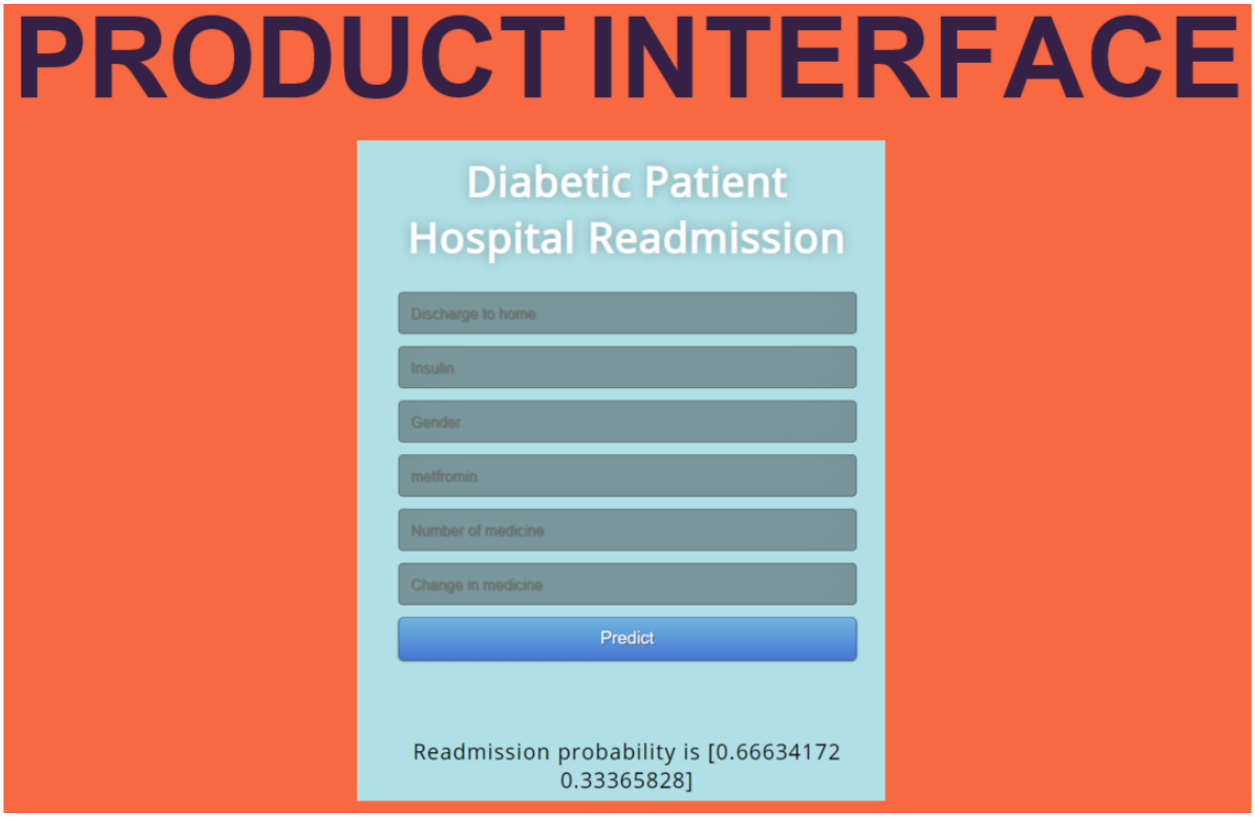


```
predictions=rf_clf.predict(X_test)
print("Predicted Result : ",predictions)

predictions = rf_clf.predict_proba(X_test)
print("Predicted Result probability : ",predictions)
```

```
Predicted Result :  [0]
Predicted Result probability :  [[0.7263008 0.2736992]]
```

- Built an interface in html for creating a complete product and mapping the deployed pickle files with 6 prominent features
- Interface has input such as Discharge to home, Insulin, Gender, metfromin, Change in medicine as binary values and Number of medicines as values ranging from 1 to 23
- Below is the image of product demo with desired inputs and result displaying probability of readmission or not



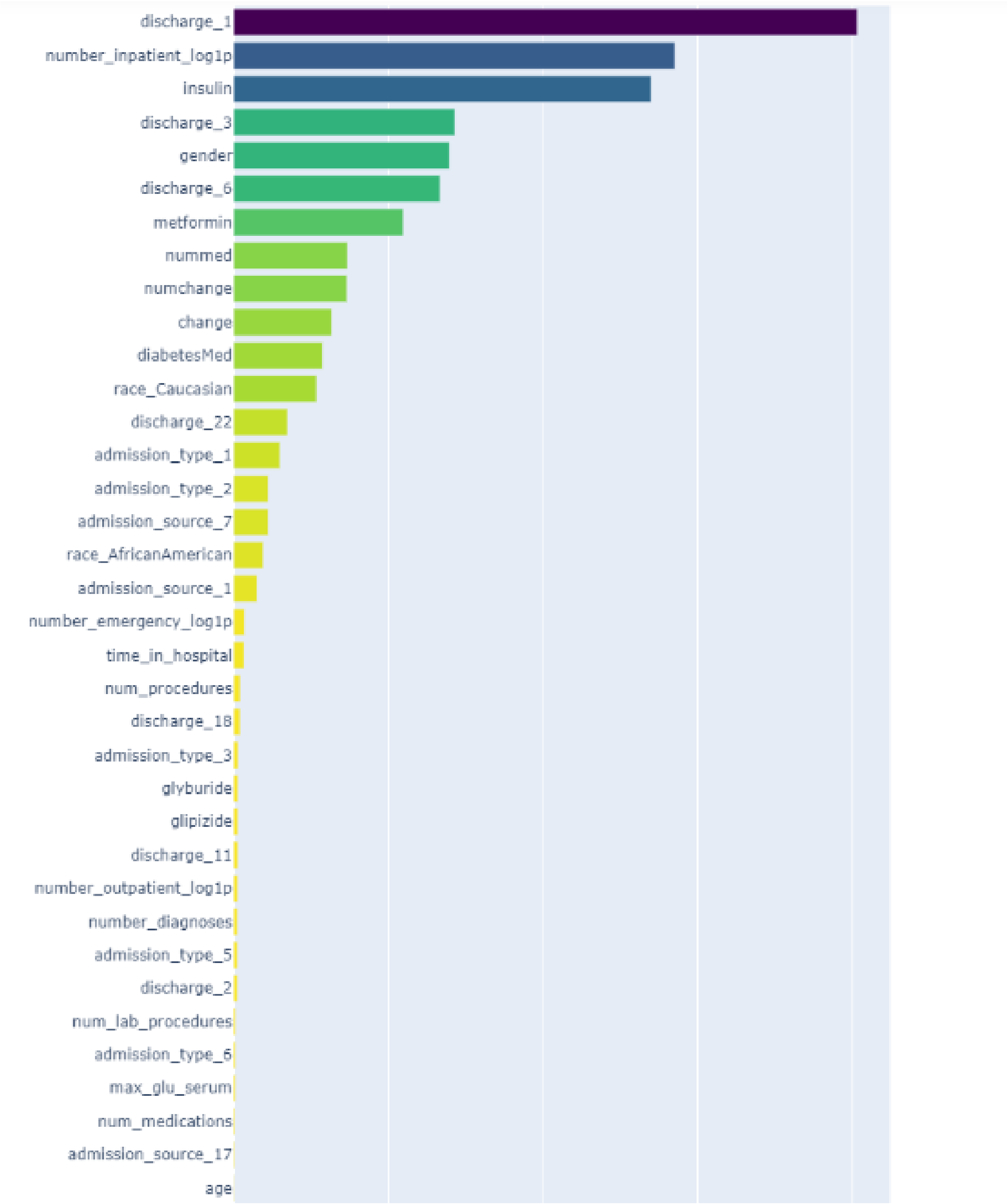
PRODUCT INTERFACE

Diabetic Patient Hospital Readmission

Readmission probability is [0.66634172 0.33365828]

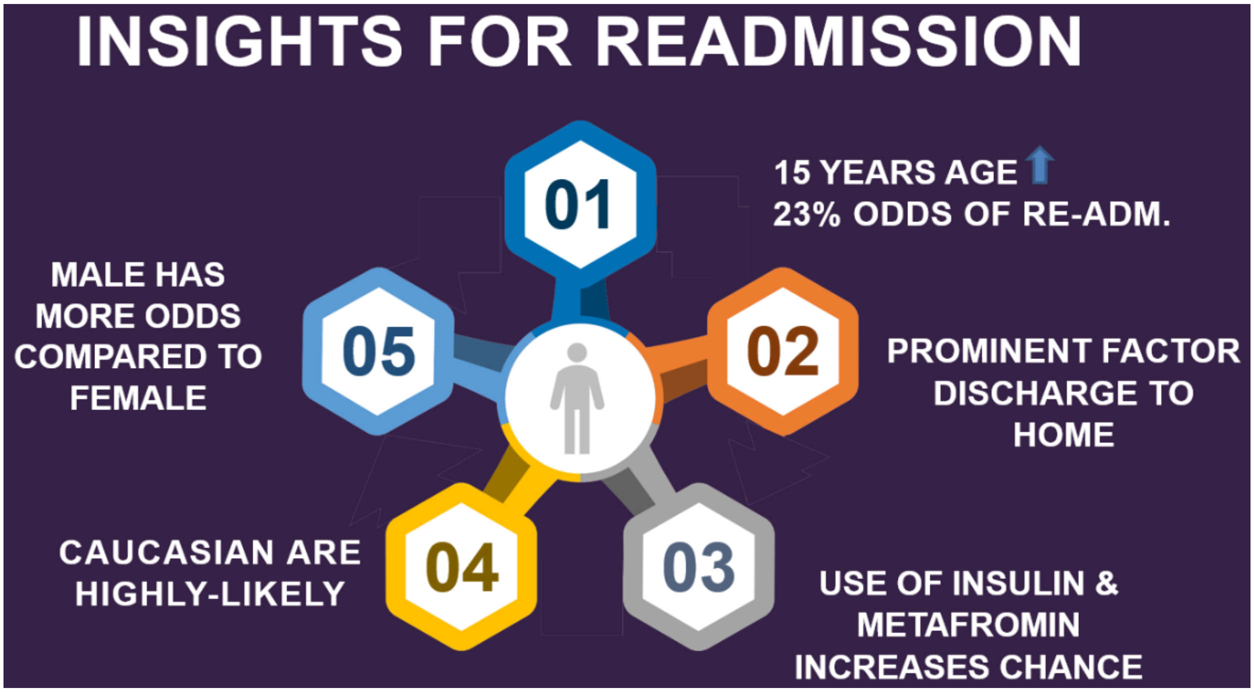
STEP: 7 - Interpretations and Insights

- Interpreting the statistics from the models and drawing insights from it
- Running the Logistic Regression to find the weightage influence of each variable
- From Random Forest, obtaining feature importance of variables and plotting bar-plot for it



INSIGHTS for Readmission:

- For every 15 years augment in age of diabetic patient there is an increase of 23% odds of readmission in hospital
- Discharge type, discharge to home is the most prominent factor in classifying readmission of diabetic patient
- Out of 23 medicine, we found that use of Insulin and Metafromin, increase the chance of readmission
- Diabetic Race Caucasian are highly-likely to get readmitted in hospital, compared to other races
- Diabetic Male has more odds compared to female for readmission in hospital



INSIGHTS FOR READMISSION

01

15 YEARS AGE ⬆
23% ODDS OF RE-ADM.

02

PROMINENT FACTOR
DISCHARGE TO
HOME

03

USE OF INSULIN &
METAFROMIN
INCREASES CHANCE

04

CAUCASIAN ARE
HIGHLY-LIKELY

05

MALE HAS
MORE ODDS
COMPARED TO
FEMALE

STEP: 8 - Improvements and Future Work

IMPROVEMENTS in project:

- Using more features such as diag_1, diag_2, diag_3 to build the machine learning model
- Adding more prominent features on product interface as input
- Available dataset was from duration 1999-2008, getting more recent data to build the model
- Since the dataset was highly imbalanced, we restricted balancing of dataset on oversampling technique. We can use under-sampling technique to balance dataset and improve the model
- Employing more machine learning algorithm to build model and check betterment, we had used only 3 algorithm Logistic Regression, Decision Tree and Random Forest

IMPROVEMENTS
01
USING MORE FEATURES IN MODEL
02
ADDING MORE STAT IMP FIELDS ON INTERFACE
03
ATTAINING CURRENT DATA
04
UTILIZING UNDER-SAMPLING TECHNIQUE
05
EMPLOYING OTHER MACHINE LEARNING METHODS

References:

- Towards Data Science:

  https://towardsdatascience.com/

- Coursera:

  https://www.coursera.org/

- Medium:

  https://medium.com/

- UCI Data repository:

  https://archive.ics.uci.edu/ml/datasets.php

- Udemy

  https://www.udemy.com/