US Diabetes Readmission Prediction

Sushant Nirantar / sniranta,

Rohan Shukla / roshukla,

Table of Contents

ABSTRACT

The project aims to study US Diabetes Readmissions and predict them for upcoming patient cases. We aim to investigate the dataset containing the various patient records and whether they had to be readmitted in the coming days. We use oversampling techniques to solve the imbalance of classes and apply various classification techniques. The metric for success is based on the recall rate since a misclassification of a critical case has more effects than otherwise.

KEYWORDS

- **SVM** – Support Vector Machine, supervised learning models used for classification and regression analysis.
- **DecisionTreeClassifier** – Commonly called as Classification Trees, models the data so that it can be classified into classes, splits on feature having maximum Info Gain.
- **AdaBoostClassifier** - Short for Adaptive Boosting, the output of weak model is combined in weighted sum that represents the final output of boosted classifier.
- **RandomForest** – Ensemble learning method used for classification and regression. For classification, the output of RandomForest is the class selected by most trees. Performance depends on data charecterstics.
- **BaggingClassifier** – Derived from Bootstrap Aggregating, is an ensemble machine learning algorithm, used to improve accuracy and stability of models. It reduces variance and helps avoid overfitting.
- **ANN** – Artificial Neural Networks are computing systems inspired by the bioloogical neural networks that constitute human brain.
- **SMOTE** – Synthetic Minority Over-sampling Technique.
- **Recall** – Fraction of relevant instances that were retrieved.
- **Precision** - Fraction of relevant instances among the retrieved instances.
- **F-Score** - Measure of test’s accuracy, calculated using precision and recall.

INTRODUCTION

Motivation

Diabetes is a precursor to many illnesses. Around 1 in Almost 10% of the cases in the United States adults are living with diabetes [1]. Also, among these, half of them were unaware of them having diabetes too [1]. In 2019, almost half million people were having diabetes, and the number is expected to rise by 25% in 2030 and 51% in 2045 [2]. Also, the lowest population in which the diabetes was undiagnosed was North America [1]. Hence, this puts America in a very good situation to utilize the knowledge and help the diabetic patients receive best care and reduce the readmission case.

Background

A high quality health care measure, hospital readmission rates are very important for cost reduction and better treatment [3]. Factors that play a key role in the risk of readmission are public insurance, lower socio economic status, and more comorbidities [3]. Factors which contributes to high readmission for patients were mainly poor health literacy and discharge process followed by post-discharge support [4]. Some of these factors are related to the background and patient's personal life, but some factors related to medical diagnosis, whose data can be collected from the patients case while admitted can be used to study and predict whether the patient is at a risk of being readmitted.

Objective and Contribution

Our aim is to use the data of existing cases of patients and predict whether a new patient is likely to be readmitted. We apply various data mining techniques and prediction models on the historical patient data. Using this knowledge, we aim to build a classifier for patients who might be needing better care and post-discharge treatment to avoid readmission. Since diabetic patients are at a higher risk of being readmitted, our study revolves around the test cases of diabetic data.

Rather focusing on the precision and accuracy we aim to contribute by building a model, which finds a balance between recall and precision of the testing cases.

Existing Work

- An improved support vector machine-based diabetic readmission prediction [5]
    - This research paper proposes an efficient way of combining the SVM and Genetic Algorithm to predict the readmission rates. This research paper uses modified SMOTE-based method to remove the data imbalance.
      The dataset had 8756 instances of patient records with 50 different attributes, and achieved the best accuracy of 81.02%.

- Predicting Hospital Readmission among Diabetics using Deep Learning [6]
    - This research paper predicts the readmission rates using Convolutional Network and data engineering combination. The paper outperforms other existing techniques on F1 Score. The data used was of 130 US hospitals from 1999 to 2008. An overall 4 improvement in F1 score and 14%5 improvement in AUC was achieved.

- Development and Validation of a Novel Tool to Predict Hospital Readmission Risk Among Patients with Diabetes [7]
    - This paper predicts the 30-day readmission among hospitalized patients with diabetes. The data instances consisted of 44203 discharge of patients, which was split into training and testing split of 60-40. Multivariate logistic regression was applied on the data and Diabetes Early Readmission Risk Indicator (DERRI™) was developed with 10 statistically significant predictors.

- Identifying Diabetic Patients with High Risk of Readmission [8]
    - Performed Machine Learning and Association rule mining methods on the 130 US Hospitals Medical Data from 1999-2008. Used Multilayer Perceptron Model, RandomForest, Adaboost and NaiveBayes to predict readmission of high risk patients and cost saved as cost per one day where doctor can run other diagnosis. The paper analysed the models under the metrics of area under the precision-recall curve.

- Impact of selected pre-processing techniques on prediction of risk of early readmission for diabetic patients in India [9]
    - This paper used 9831 test cases of patients diagnosed with diabetes and applied logistic regression, Naïve Bayes and C4.5 Decision Tree algorithms. They pre-processed the data and solved data imbalance by using a hybrid approach of over-sampling and under-sampling techniques. The major achievement of the paper included increasing the recall rate from 0.02-0.23 to 0.78-0.85.

- Prediction on diabetes patient's hospital readmission rates [10]
    - Used feature scaling and feature encoding followed by oversampling to solve the data imbalance error. Features contributing most towards readmission were identified using decision tree and random forest. Also, logistic regression, adaboost and XGBoost were applied to achieve significant accuracy and recall rates.

PROCESS

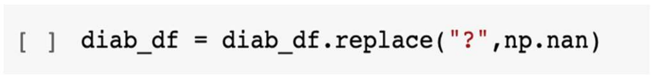
Analysis of Data

The dataset used for studying and modeling was sourced from Center for Clinical and Translational Research, Virginia Commonwealth University. It represents 10 years (1999-2008) of clinical care at 130 US Hospitals and integrated delivery networks. The recorded information is Health Insurance Portability and Accountability Act (HIPAA) compliant.

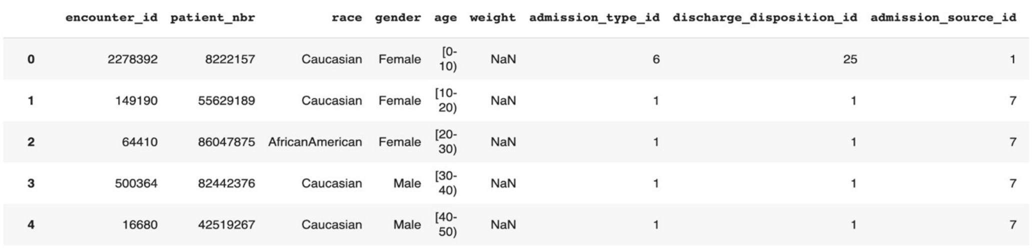
Data Pre-Processing

The dataset contains over 100,000 records of patient-provider interaction that is captured using 50 features which are used to describe simple attributes like name, weight, age to Diabetes specific attributes like Hb1AC result, metformin, insulin, glipizide, which are categorical variables used to describe the result of test, dose of medicine (Stable/Up/No/Down), number of days spent in hospital, number of tests, number of lab tests, number of diabetes medicines. Every feature is constrained to accept non-null values only. The type of data that each attribute holds is either int64 or object.

On careful examination of the dataset, we found that null/missing values in the dataset are represented by “?”. Our first task was to replace “?” with np.nan and find the percentage of missing values for every feature.



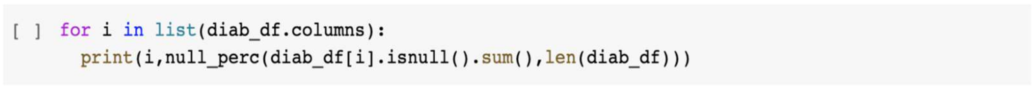
```
[ ]  diab_df = diab_df.replace("?",np.nan)
```



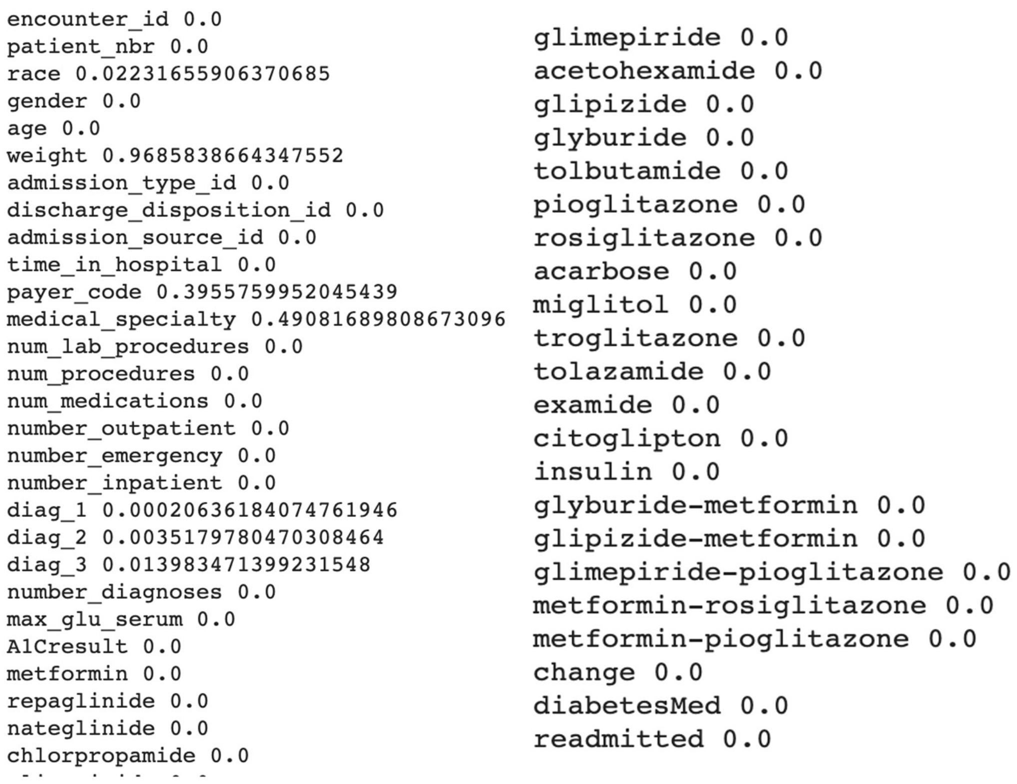
	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id
0	2278392	8222157	Caucasian	Female	[0-10)	NaN	6	25	1
1	149190	55629189	Caucasian	Female	[10-20)	NaN	1	1	7
2	64410	86047875	AfricanAmerican	Female	[20-30)	NaN	1	1	7
3	500364	82442376	Caucasian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

This helps us to more accurately calculate the percentage of missing values for each feature. We find the same and using principles of Data Mining, we will either discard the feature or fill the missing value with some value.

Now we will find the percentage of missing/NaN values for each feature.



```
[ ] for i in list(diab_df.columns):
      print(i,null_perc(diab_df[i].isnull().sum(),len(diab_df)))
```



```
encounter_id 0.0
patient_nbr 0.0
race 0.02231655906370685
gender 0.0
age 0.0
weight 0.9685838664347552
admission_type_id 0.0
discharge_disposition_id 0.0
admission_source_id 0.0
time_in_hospital 0.0
payer_code 0.3955759952045439
medical_specialty 0.49081689808673096
num_lab_procedures 0.0
num_procedures 0.0
num_medications 0.0
number_outpatient 0.0
number_emergency 0.0
number_inpatient 0.0
diag_1 0.00020636184074761946
diag_2 0.0035179780470308464
diag_3 0.013983471399231548
number_diagnoses 0.0
max_glu_serum 0.0
A1Cresult 0.0
metformin 0.0
repaglinide 0.0
nateglinide 0.0
chlorpropamide 0.0
```

```
glimepiride 0.0
acetohexamide 0.0
glipizide 0.0
glyburide 0.0
tolbutamide 0.0
pioglitazone 0.0
rosiglitazone 0.0
acarbose 0.0
miglitol 0.0
troglitazone 0.0
tolazamide 0.0
examide 0.0
citoglipton 0.0
insulin 0.0
glyburide-metformin 0.0
glipizide-metformin 0.0
glimepiride-pioglitazone 0.0
metformin-rosiglitazone 0.0
metformin-pioglitazone 0.0
change 0.0
diabetesMed 0.0
readmitted 0.0
```

From the above results we can clearly see that weight, payer_code and medical_speciality have significant Null values, the percentage of these values are more than what is acceptable when going by Data Mining Principles, So we discard these features.

```
clean_diab_df=diab_df.drop(columns=['weight','payer_code','medical_specialty'])
```

	encounter_id	patient_nbr	race	gender	age	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

We observe that the “age” feature has range values, since we intend to use this dataset for training a model, we change the type of “age” to numeric and fill it with the mean of the interval associated with the record.

```
clean_diab_df['age']=clean_diab_df['age'].replace({"[0-10)":5,"[10-20)":15,"[20-30)":25,"[30-40)":35,"[40-50)":45,"[50-60)":55,"[60-70)":
```

	race	gender	age	diag_1	diag_2	diag_3	max_glu_serum	A1Cresult	metformin	repaglinide	nateglinide	chlorpropamide
0	Caucasian	Female	5	250.83	276	250	None	None	No	No	No	No
1	Caucasian	Female	15	276	250.01	255	None	None	No	No	No	No
2	AfricanAmerican	Female	25	648	250	V27	None	None	No	No	No	No
3	Caucasian	Male	35	8	250.43	403	None	None	No	No	No	No
4	Caucasian	Male	45	197	157	250	None	None	No	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...
101761	AfricanAmerican	Male	75	250.13	291	458	None	>8	Steady	No	No	No
101762	AfricanAmerican	Female	85	560	276	787	None	None	No	No	No	No
101763	Caucasian	Male	75	38	590	296	None	None	Steady	No	No	No
101764	Caucasian	Female	85	996	285	998	None	None	No	No	No	No
101765	Caucasian	Male	75	530	530	787	None	None	No	No	No	No

For the remaining features that had negligible missing values, we replace them with the mode of that particular feature.

```
clean_diab_df['race']=clean_diab_df['race'].fillna(clean_diab_df['race'].mode()[0])
```

```
clean_diab_df['diag_1']=clean_diab_df['diag_1'].fillna(clean_diab_df['diag_1'].mode()[0])
```

```
clean_diab_df['diag_2']=clean_diab_df['diag_2'].fillna(clean_diab_df['diag_2'].mode()[0])
```

```
clean_diab_df['diag_3']=clean_diab_df['diag_3'].fillna(clean_diab_df['diag_3'].mode()[0])
```

Checking again for missing values for each feature.

```python
clean_diab_df.isnull().sum()
```

```
encounter_id                  0
patient_nbr                   0
race                          0
gender                        0
age                           0
admission_type_id             0
discharge_disposition_id      0
admission_source_id           0
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
max_glu_serum                 0
A1Cresult                     0
metformin                     0
repaglinide                   0
nateglinide                   0
chlorpropamide                0
glipizide                     0
glyburide                     0
tolbutamide                   0
pioglitazone                  0
rosiglitazone                 0
acarbose                      0
miglitol                      0
troglitazone                  0
tolazamide                    0
examide                       0
citoglipton                   0
insulin                       0
glyburide-metformin           0
glipizide-metformin           0
glimepiride-pioglitazone      0
metformin-rosiglitazone       0
metformin-pioglitazone        0
change                        0
diabetesMed                   0
readmitted                    0
```

The target feature for our dataset is “readmitted” which has three possible classes namely, “NO”, “>30” and “<30”. For our model training we have taken two different cases, first we train our models using the same classes provided in the dataset and second, we merge the “>30” and “NO” readmission classes as one and “<30” as another. The idea behind this is, a readmission of “>30” days is very vague and can lead to providing specialized care to a patient who may not need such specialized care. We first see the distribution of all three classes in the dataset.

```python
diab_df['readmitted'].value_counts()
```

```
NO      54864
>30     35545
<30     11357
Name: readmitted, dtype: int64
```

Merging “>30” and “NO” readmission classes for EDA.

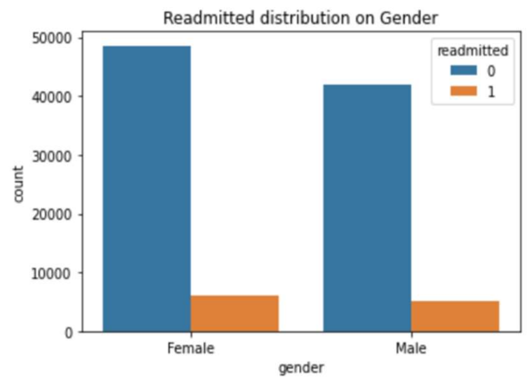
```python
diab_df['readmitted']=diab_df['readmitted'].replace({"NO":0,">30":0,"<30":1})
```

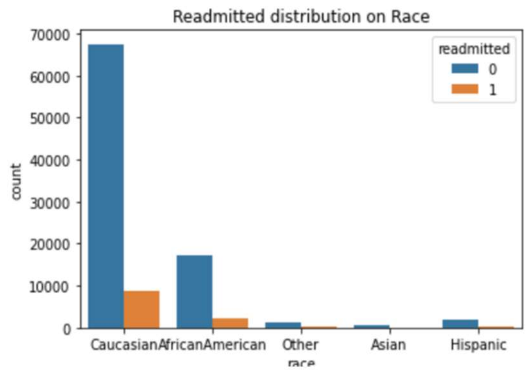
Distribution of the “new” target classes in the dataset.

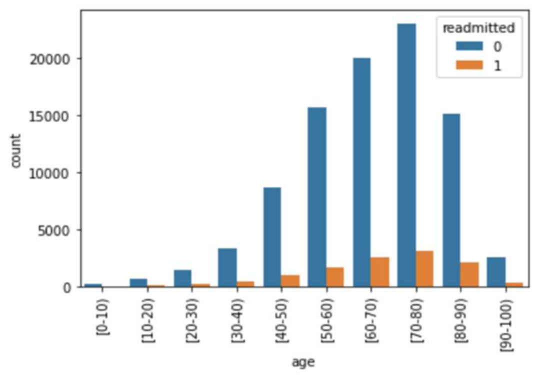
```python
plt.hist(diab_df['readmitted'])
```

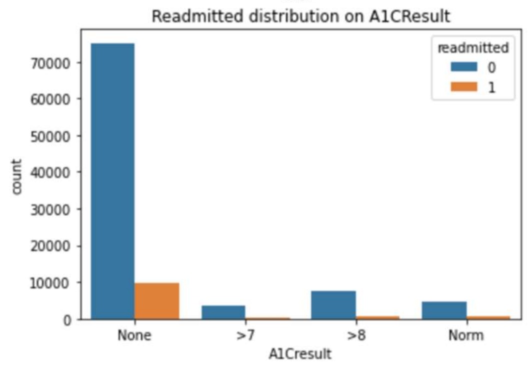
Exploratory Data Analysis

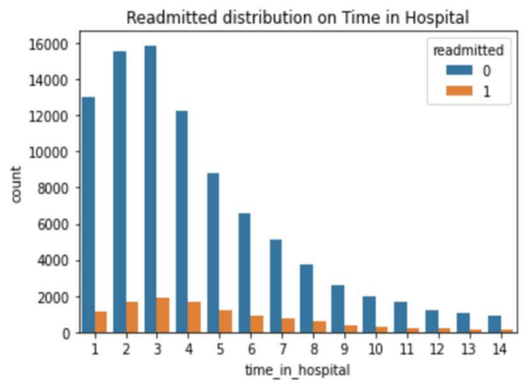
We now proceed to visualize the distribution of target feature “readmitted” against every feature of the dataset.

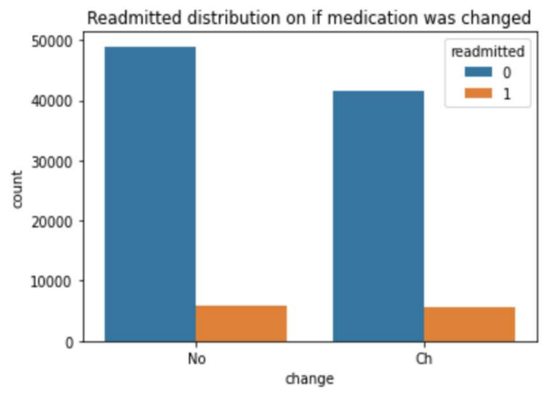


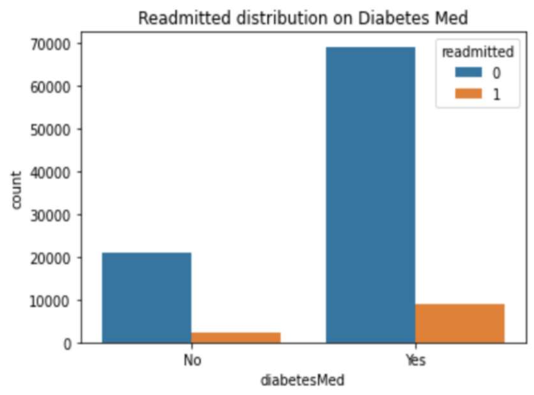


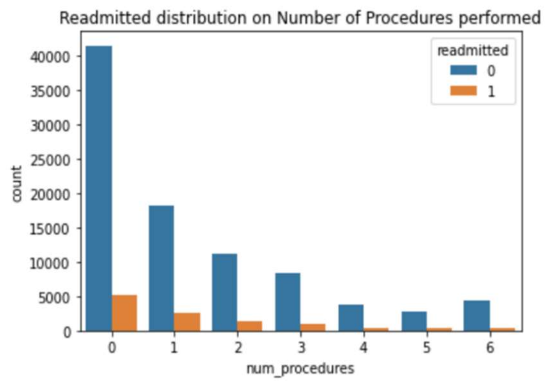


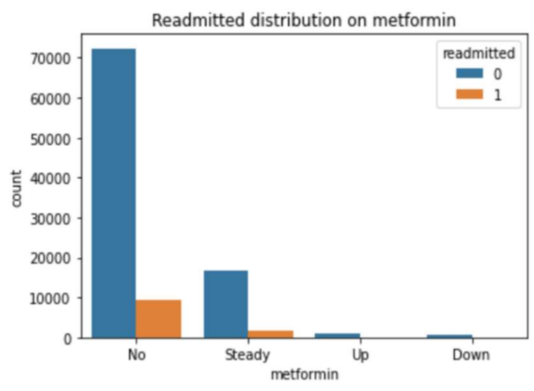




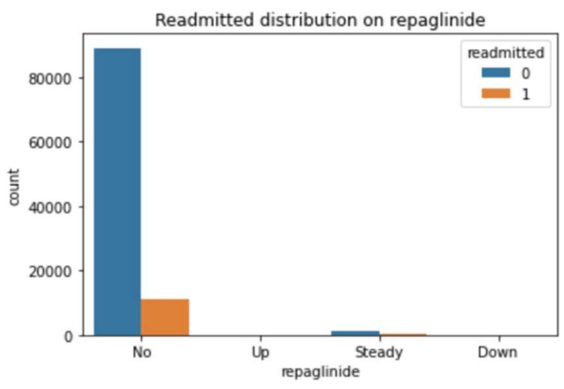




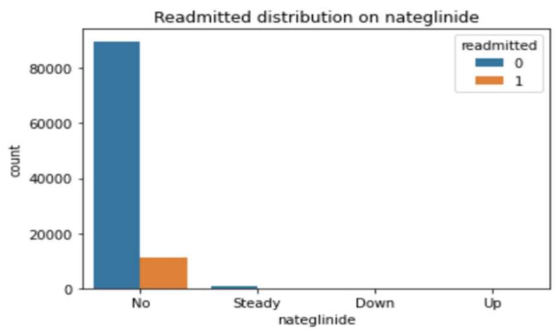




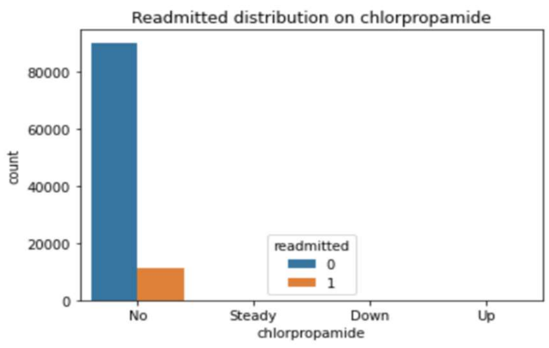
Readmitted distribution on metformin
count
metformin
readmitted
0
1
No
Steady
Up
Down



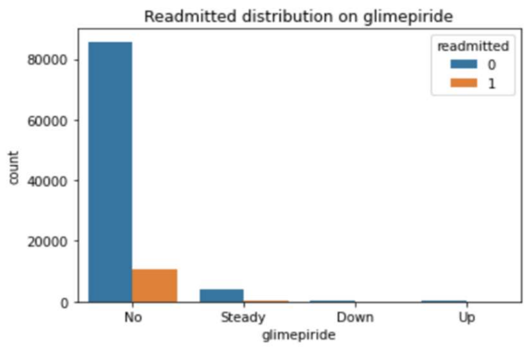
Readmitted distribution on repaglinide
count
repaglinide
readmitted
0
1
No
Up
Steady
Down



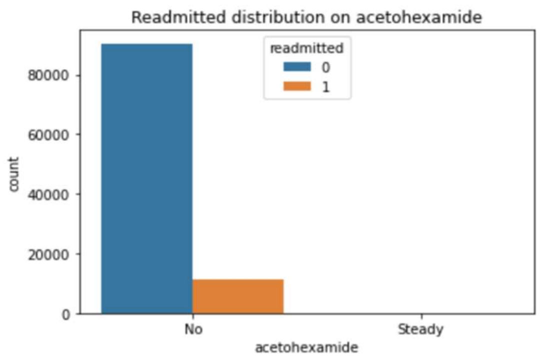
Readmitted distribution on nateglinide
count
nateglinide
readmitted
0
1
No
Steady
Down
Up



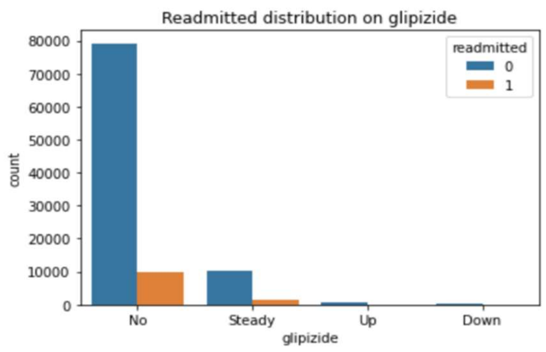
Readmitted distribution on chlorpropamide
count
chlorpropamide
readmitted
0
1
No
Steady
Down
Up



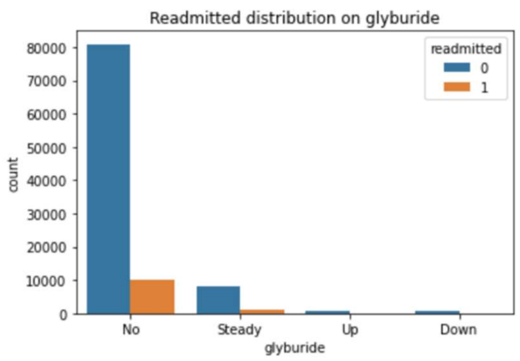
Readmitted distribution on glimepiride
count
glimepiride
readmitted
0
1
No
Steady
Down
Up



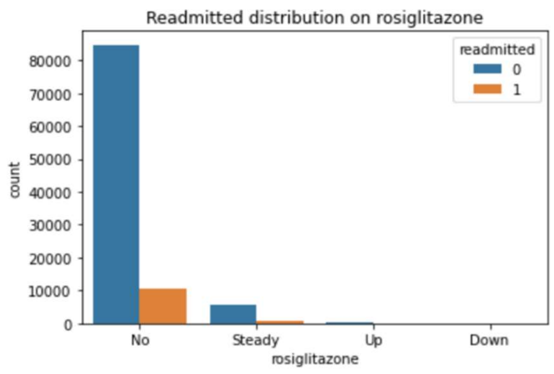
Readmitted distribution on acetohexamide
count
acetohexamide
readmitted
0
1
No
Steady

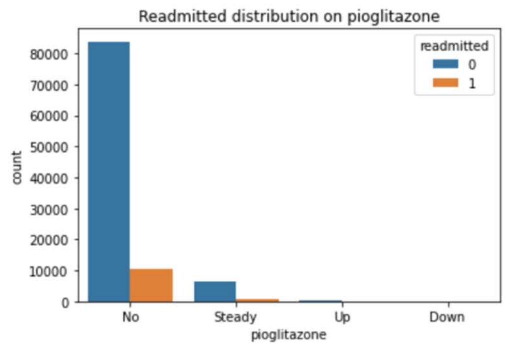


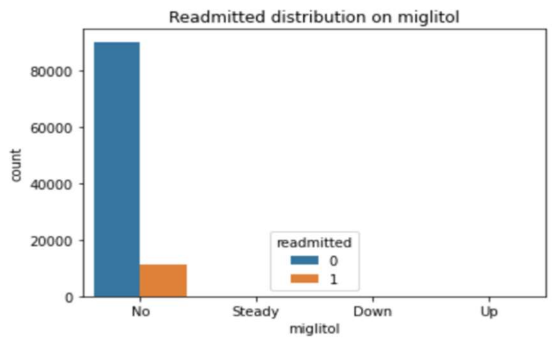
Readmitted distribution on glipizide
count
glipizide
readmitted
0
1
No
Steady
Up
Down

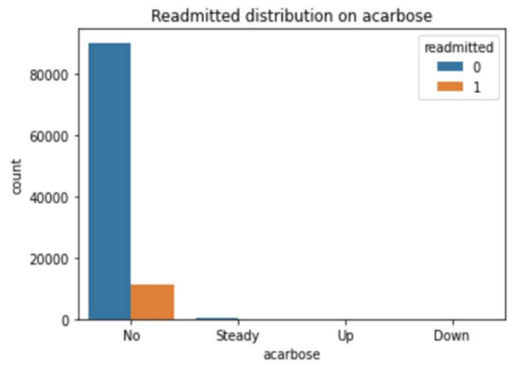


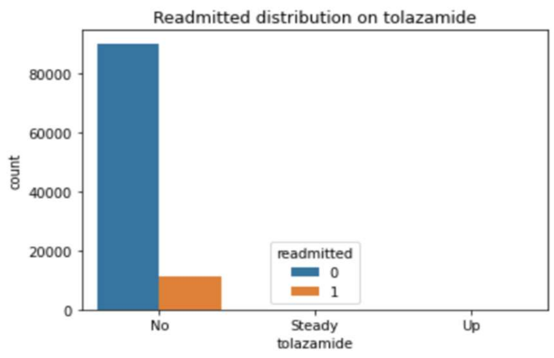
Readmitted distribution on glyburide
count
glyburide
readmitted
0
1
No
Steady
Up
Down

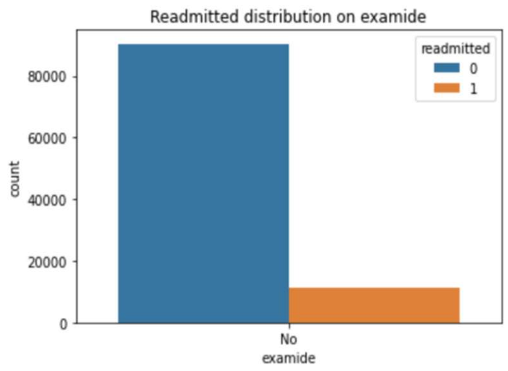

Readmitted distribution on rosiglitazone
readmitted
0
1
count
80000
70000
60000
50000
40000
30000
20000
10000
0
No
Steady
Up
Down
rosiglitazone

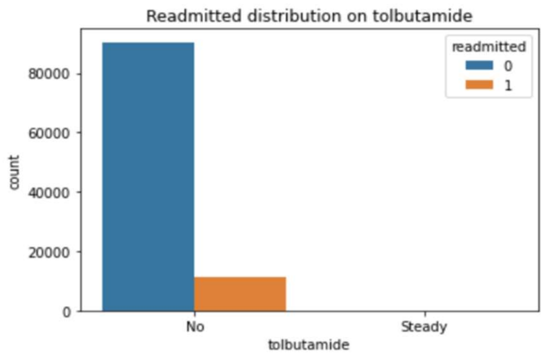

Readmitted distribution on pioglitazone
readmitted
0
1
count
80000
70000
60000
50000
40000
30000
20000
10000
0
No
Steady
Up
Down
pioglitazone

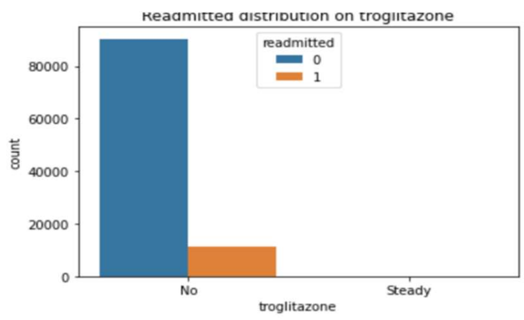

Readmitted distribution on miglitol
readmitted
0
1
count
80000
60000
40000
20000
0
No
Steady
Down
Up
miglitol

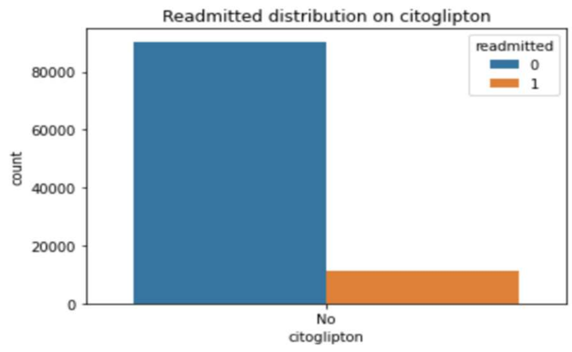

Readmitted distribution on acarbose
readmitted
0
1
count
80000
60000
40000
20000
0
No
Steady
Up
Down
acarbose

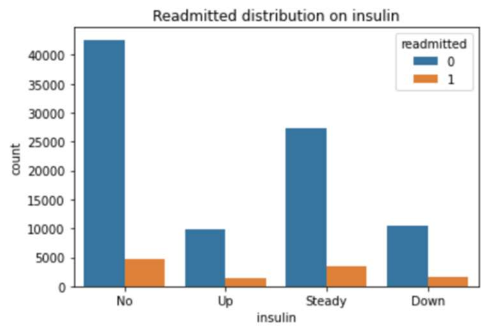

Readmitted distribution on tolazamide
readmitted
0
1
count
80000
60000
40000
20000
0
No
Steady
Up
tolazamide

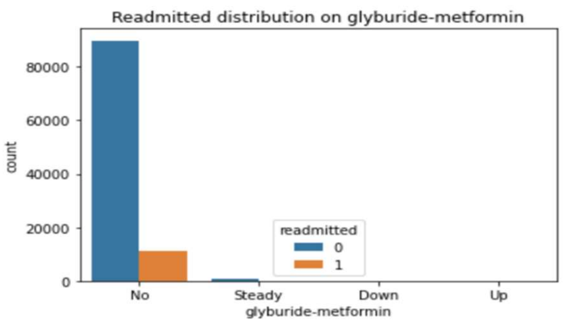

Readmitted distribution on examide
readmitted
0
1
count
80000
60000
40000
20000
0
No
examide

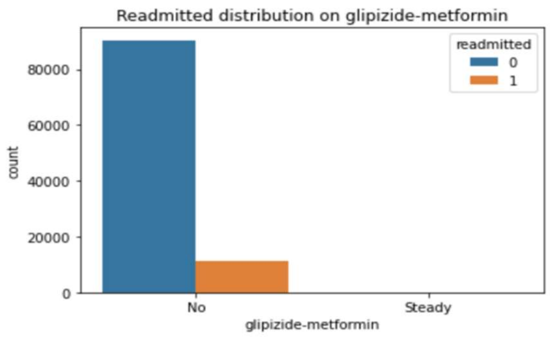

Readmitted distribution on tolbutamide
readmitted
0
1
count
80000
60000
40000
20000
0
No
Steady
tolbutamide

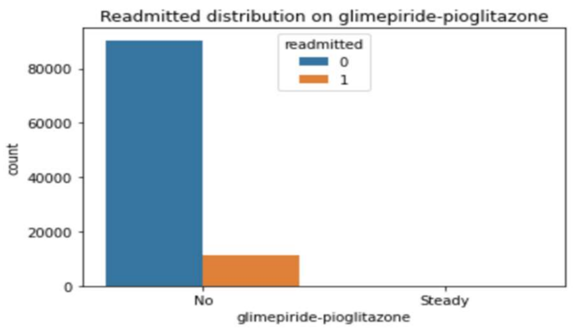

Readmitted distribution on troglitazone
readmitted
0
1
count
80000
60000
40000
20000
0
No
Steady
troglitazone

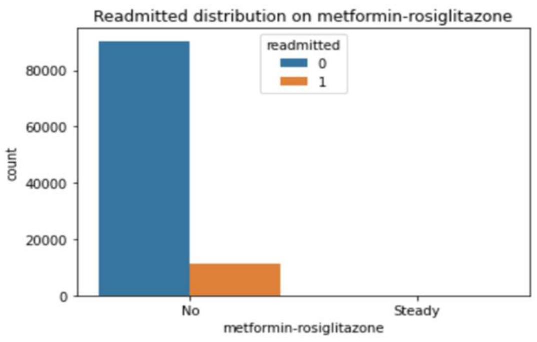

Readmitted distribution on citoglipton
readmitted
0
1
count
80000
60000
40000
20000
0
No
citoglipton

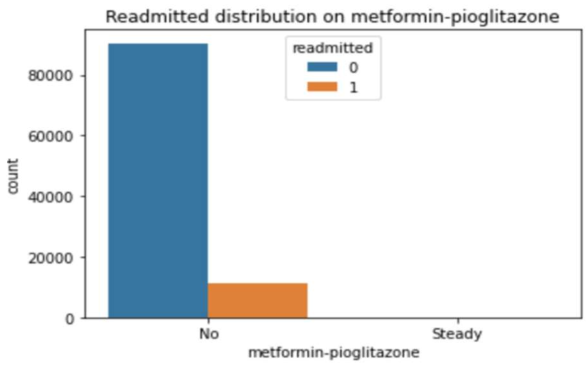

Readmitted distribution on insulin
readmitted
0
1
count
40000
35000
30000
25000
20000
15000
10000
5000
0
No
Up
Steady
Down
insulin

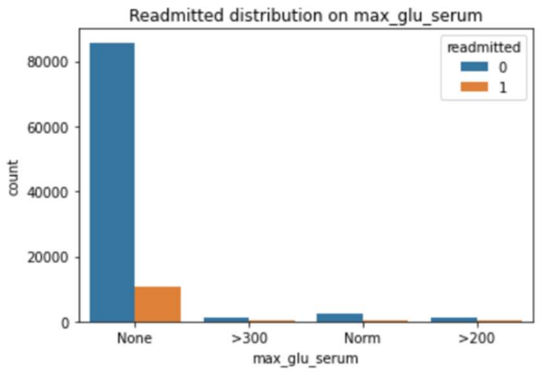












This is one of the examples of failed maps. It is a non-interactive chart. Although the colour variants are very effective, one cannot hover over it and get the details.

Modelling

After completing the data cleaning and preprocessing, we move to building a model for the given dataset such that it is able to correctly classify the given information of the patient into being readmitted or not readmitted. We use DecisionTreeClassifier, RandomForrest, AdaBoostClassifier, BaggingClassifier and ANN for building our model. We chose these classifiers because we want to include the categorical data features to be part of the decision making process. Since we are working with some categorical data features, we use LabelEncoder. After performing this step, we remove the target feature, “readmitted” from the data frame and split the data into train and test data set in ratio 70:30. We then study the performance of each model and compare them with each other.

```
label_encoder=LabelEncoder()
for i in object_data1:
  object_data1[i]=label_encoder.fit_transform(object_data1[i].astype('str'))
```

The above code was used to transform the object data types into classes with values 0 to (number of classes -1).

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)
```

Fitting a DecisionTreeClassifier model to the above dataset.

```
dtree=DecisionTreeClassifier(max_depth=25)
dtree.fit(x_train,y_train)
```

Examining the accuracy of the model.

```
y_pred1=dtree.predict(x_train)
```

```
metrics.accuracy_score(y_train,y_pred1)
```

```
0.9693966364376562
```

```
y_pred=dtree.predict(x_test)
```

```
metrics.accuracy_score(y_test,y_pred)
```

```
0.8254446591765207
```

The metric for evaluation for our work is the recall rate for the people readmitted to hospital within 30 days. This can be accomplished using classification report.

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

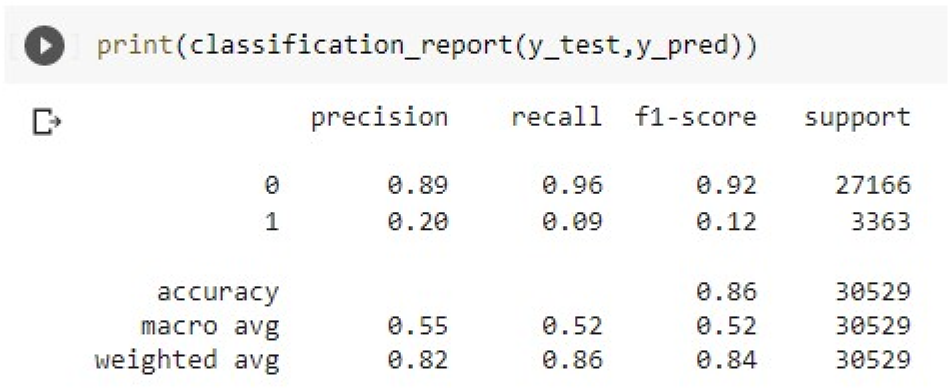
```
              precision    recall  f1-score   support

           0       0.89      0.91      0.90     27056
           1       0.18      0.15      0.16      3473

    accuracy                           0.83     30529
   macro avg       0.54      0.53      0.53     30529
weighted avg       0.81      0.83      0.82     30529
```

We see that the recall rate for our model is not that great and the medical field requires this recall rate to be high as classifying a patient with a good chance of readmission as the opposite is potentially harmful to the health of the patient. This happens because the number of records classified as “<30” is very less as compared to “NO” and “>30”. To address this problem we used SMOTE technique to synthetically over sample the minority class, in our case the class “<30”. We will get back to SMOTE sampling ahead, for now we examine the performance of RandomForrest, AdaBoostClassifier and BaggingClassifier.

Fitting RandomForrestClassifier model to the dataset.

```
rfc=RandomForestClassifier(n_estimators=150)
rfc.fit(x_train,y_train)
y_pred=rfc.predict(x_test)
```

Evaluating the accuracy of the model.

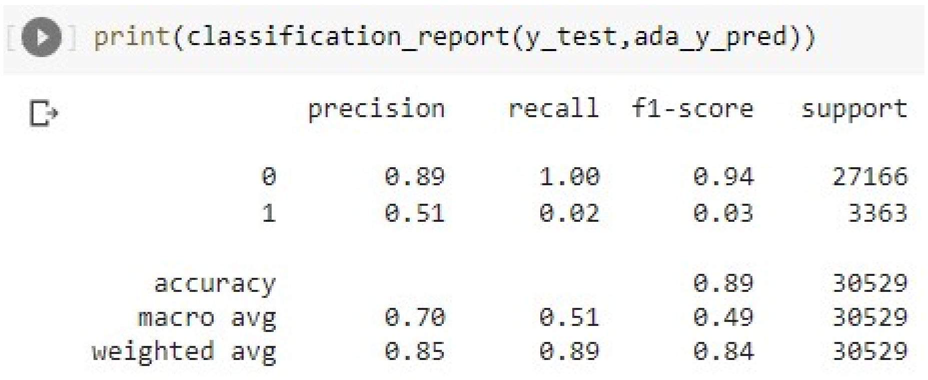


```
print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.89      0.96      0.92     27166
           1       0.20      0.09      0.12      3363

    accuracy                           0.86     30529
   macro avg       0.55      0.52      0.52     30529
weighted avg       0.82      0.86      0.84     30529
```

Fitting AdaBoostClassifier model to the dataset.

```
ada_model=AdaBoostClassifier()
ada_model.fit(x_train,y_train)
```

Evaluating the accuracy of the model.



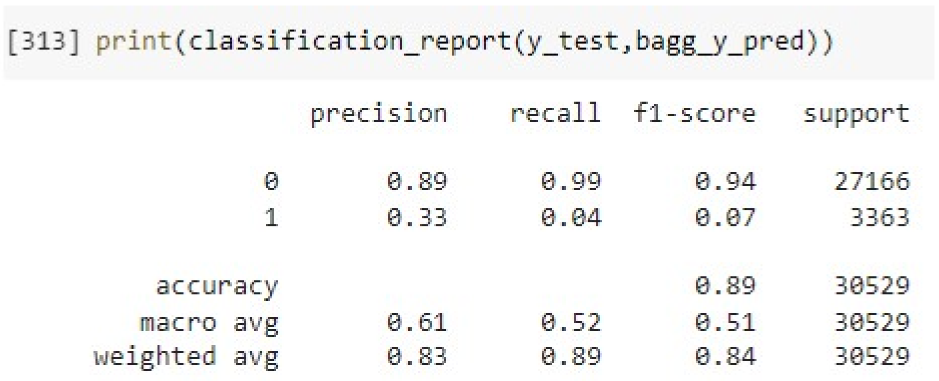
```
print(classification_report(y_test,ada_y_pred))

              precision    recall  f1-score   support

           0       0.89      1.00      0.94     27166
           1       0.51      0.02      0.03      3363

    accuracy                           0.89     30529
   macro avg       0.70      0.51      0.49     30529
weighted avg       0.85      0.89      0.84     30529
```

Fitting BaggingClassifier model to the dataset.

```
bagg_model=BaggingClassifier()
```

```
bagg_model.fit(x_train,y_train)
```

Evaluating the accuracy of the model.



```
[313] print(classification_report(y_test,bagg_y_pred))

              precision    recall  f1-score   support

           0       0.89      0.99      0.94     27166
           1       0.33      0.04      0.07      3363

    accuracy                           0.89     30529
   macro avg       0.61      0.52      0.51     30529
weighted avg       0.83      0.89      0.84     30529
```

Coming back to the problem of recall, we address it using SMOTE oversampling technique, in which the distribution of minority class is synthetically increased to match the majority class so that the model can better fit the data.

```
train.rename(columns={44:'readmitted'}, inplace=True)
```

```
not_readm = train[train.readmitted==0]
readm = train[train.readmitted==1]
```

```
from sklearn.utils import resample
read_upsampled = resample(readm,
                          replace=True, # sample with replacement
                          n_samples=len(not_readm), # match number in majority class
                          random_state=27) # reproducible results
```

```
upsampled = pd.concat([not_readm, read_upsampled])
```

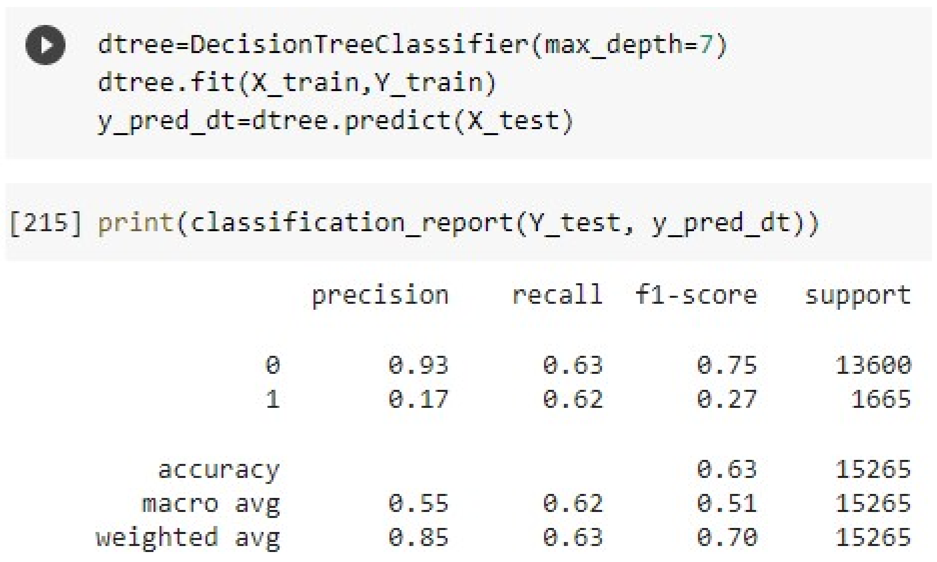
```
upsampled.readmitted.value_counts()
```

```
1    63215
0    63215
Name: readmitted, dtype: int64
```

```
Y_train = upsampled.readmitted
X_train = upsampled.drop('readmitted', axis=1)
```

We can see that by using SMOTE technique, the distribution of “readmitted” and “not readmitted” classes is equal. We use this data to train the above mentioned models and evaluate their performance again.

Fitting and evaluating DecisionTreeClassifier on “new” SMOTE dataset.

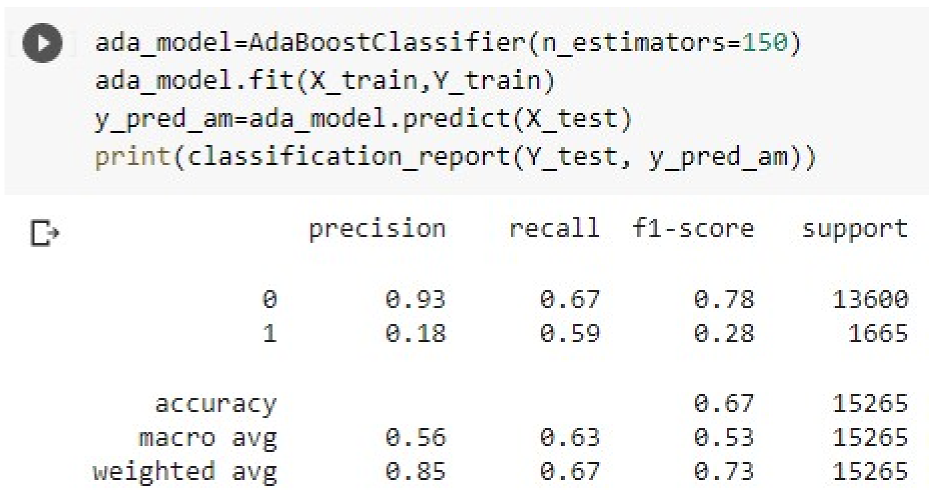


```
dtree=DecisionTreeClassifier(max_depth=7)
dtree.fit(X_train,Y_train)
y_pred_dt=dtree.predict(X_test)
```

```
[215] print(classification_report(Y_test, y_pred_dt))
```

```
              precision    recall  f1-score   support

           0       0.93      0.63      0.75     13600
           1       0.17      0.62      0.27      1665

    accuracy                           0.63     15265
   macro avg       0.55      0.62      0.51     15265
weighted avg       0.85      0.63      0.70     15265
```

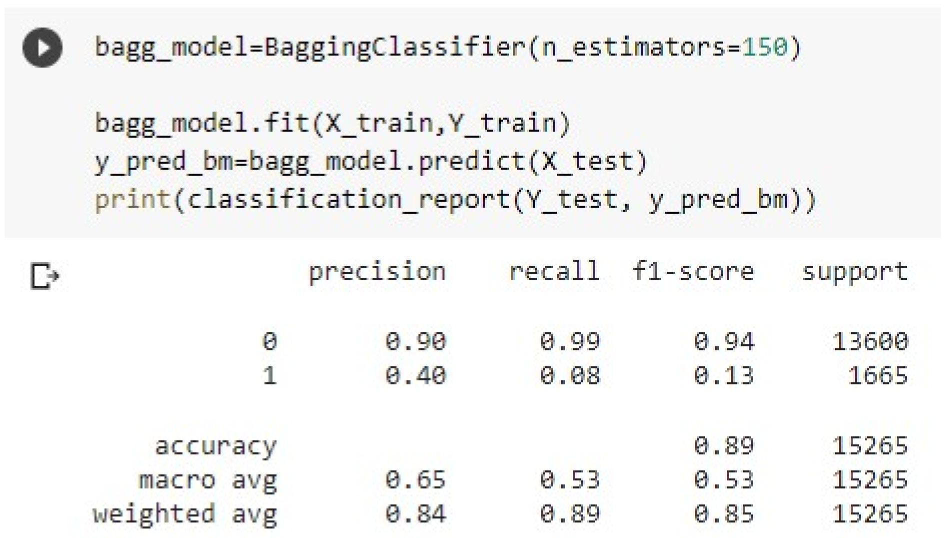
Fitting and evaluating AdaBoostClassifier on “new” SMOTE dataset.



```
ada_model=AdaBoostClassifier(n_estimators=150)
ada_model.fit(X_train,Y_train)
y_pred_am=ada_model.predict(X_test)
print(classification_report(Y_test, y_pred_am))
```

```
              precision    recall  f1-score   support

           0       0.93      0.67      0.78     13600
           1       0.18      0.59      0.28      1665

    accuracy                           0.67     15265
   macro avg       0.56      0.63      0.53     15265
weighted avg       0.85      0.67      0.73     15265
```

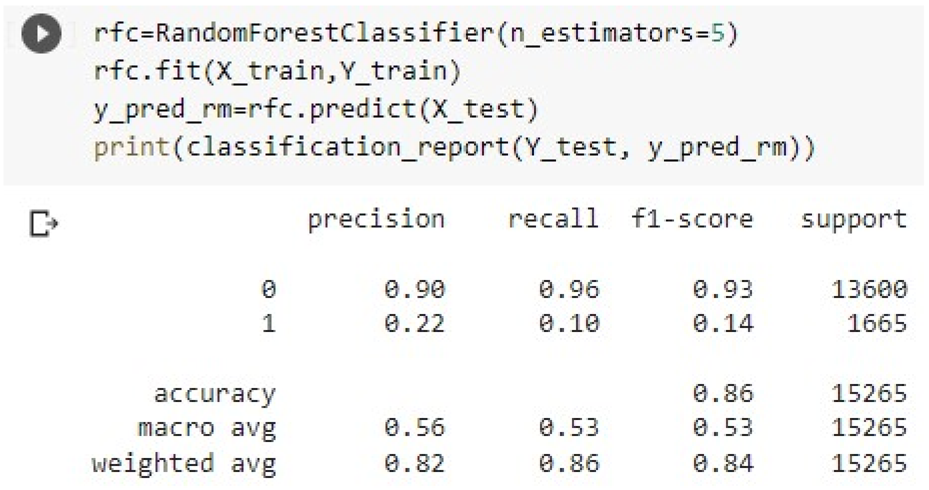
Fitting and evaluating BaggingClassifier on “new” SMOTE dataset.



```
bagg_model=BaggingClassifier(n_estimators=150)

bagg_model.fit(X_train,Y_train)
y_pred_bm=bagg_model.predict(X_test)
print(classification_report(Y_test, y_pred_bm))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94     13600
           1       0.40      0.08      0.13      1665

    accuracy                           0.89     15265
   macro avg       0.65      0.53      0.53     15265
weighted avg       0.84      0.89      0.85     15265
```

Fitting and evaluating RandomForrestClassifier on “new” SMOTE dataset.



```
rfc=RandomForestClassifier(n_estimators=5)
rfc.fit(X_train,Y_train)
y_pred_rm=rfc.predict(X_test)
print(classification_report(Y_test, y_pred_rm))
```

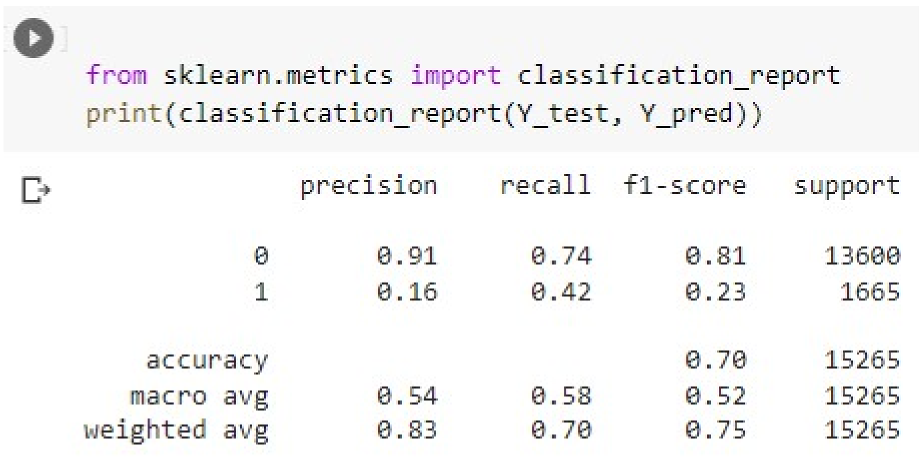
```
              precision    recall  f1-score   support

           0       0.90      0.96      0.93     13600
           1       0.22      0.10      0.14      1665

    accuracy                           0.86     15265
   macro avg       0.56      0.53      0.53     15265
weighted avg       0.82      0.86      0.84     15265
```

Implementing ANN to the above dataset.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
model = Sequential([
    Dense(256,activation='relu', input_shape=(44,)),

    Dense(256,activation='relu'),

    Dense(128,activation='relu'),

    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid'),
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
hist = model.fit(X_train, Y_train,
          batch_size=128, epochs=50,
          validation_data=(X_val, Y_val))
```

Evaluating the performance of ANN.



```
from sklearn.metrics import classification_report
print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.74      0.81     13600
           1       0.16      0.42      0.23      1665

    accuracy                           0.70     15265
   macro avg       0.54      0.58      0.52     15265
weighted avg       0.83      0.70      0.75     15265
```

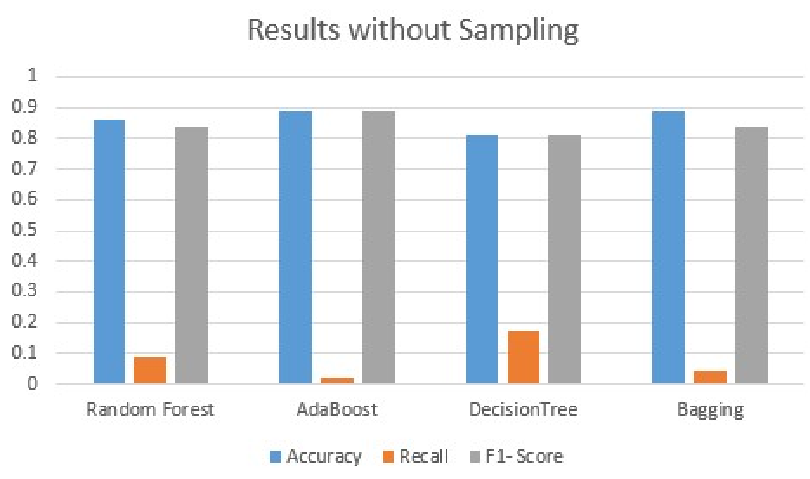
RESULTS AND INSIGHTS

The model was run on various parameters and the results were obtained.

First, the vanilla classification algorithms were run on the pre-processed data, where the dataset was used without solving the data imbalance, and with two data classes.

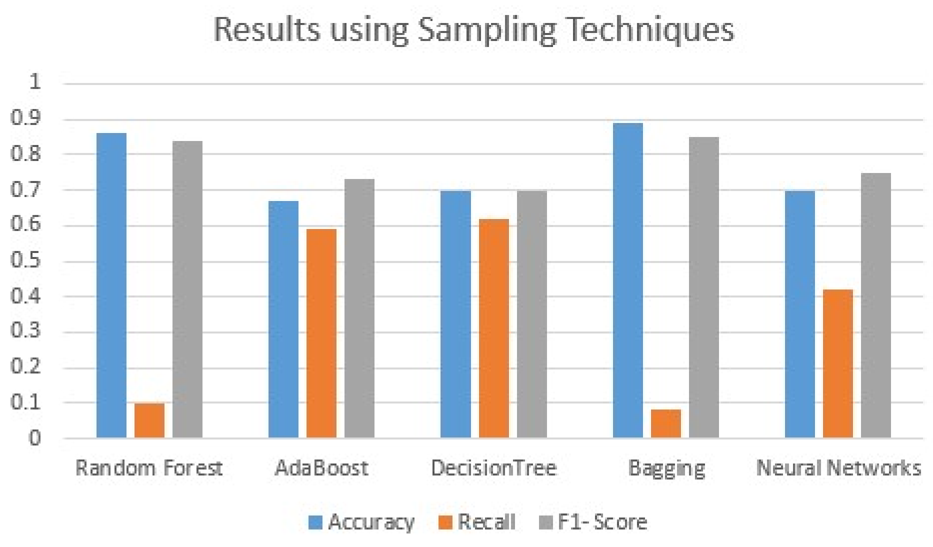
The results of the models run were:

	Accuracy	**Recall**	**F1-Score**
Random Forest	0.86	0.09	0.84
AdaBoost	0.89	0.02	0.89
DecisionTree	0.81	0.17	0.81
Bagging	0.89	0.04	0.84

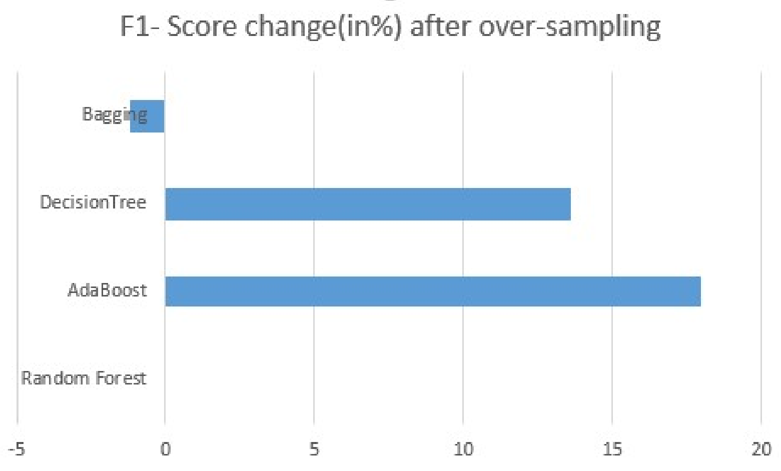


Since, the recall rates were very less, the model was the oversampled using SMOTE, and the results obtained were:

	Accuracy	Recall	F1-Score
Random Forest	0.86	0.1	0.84
AdaBoost	0.67	0.59	0.73
DecisionTree	0.7	0.62	0.7
Bagging	0.89	0.08	0.85
Neural Networks	0.7	0.42	0.75



Here, we can notice that the recall rates of the models have increased significantly for the algorithms. The F1- Score increased significantly in two of the algorithms.



CONCLUSION

Diabetes is a metabolic disorder characterized by high sugar level over prolonged period of time. In 2019 almost half million people have been diagnosed with diabetes, with type-2 diabetes sharing over 90% of the total cases. This number is expected to grow over the coming years. Diabetes if left untreated may cause serious complications and may even result in early death of the patient. Our project studied the dataset provided by the Virginia Commonwealth University, that describes diabetic patient data from 1999-2008. We specifically aimed to build a model using classification techniques that we learned during the coursework, so that we are able to classify patient information given as being readmitted or not readmitted. We focused on readmission because, diabetes being a precursor to many diseases cannot be left untreated, and providing better care to the patient may result in healthier society and will also reduce the per capita spending on healthcare. Since every patient is different, the care given to them also needs to be different and so we aimed to provide a better feedback system so that better care can be given during the hospital stay and readmission of patient within 30 days is avoided. We built our models on DecisionTree, AdaBoostClassifier, BaggingClassifier, RandomForrest and ANN. The proposed methodology, used oversampling technique on the misclassified readmission target label, and then achieved better recall rates when the above mentioned models were run. The model achieved a rise in 13%-17% of the recall of readmissions in 30 days.

FUTURE WORK

Our current work focuses on building a model to predict that readmission of patient based on data provided to it. In medical decision making system, we are more concerned with classifying a True Positive as Positive rather than negative i.e. A diabetic patient with a chance of readmission within 30 days must be flagged as such only and not as not readmitted. We tackle this problem by using SMOTE oversampling technique, as the number of "1" class records in the dataset were very few as compared to the "0" class. Using this, we were able to get a recall rate of almost 62% and an accuracy of almost 70%. Oversampling requires a delicate balance, which can be explored in future studies to improve the recall and precision of the model even further. Furthermore, under-sampling of majority target label, or a hybrid of these could also be used to achieve better results.

REFERENCES

[1] Ogurtsova, K., Guariguata, L., Barengo, N. C., Ruiz, P. L. D., Sacre, J. W., Karuranga, S., ... & Magliano, D. J. (2021). IDF Diabetes Atlas: Global estimates of undiagnosed diabetes in adults for 2021. Diabetes Research and Clinical Practice, 109118.

[2] Saeedi, P., Petersohn, I., Salpea, P., Malanda, B., Karuranga, S., Unwin, N., ... & IDF Diabetes Atlas Committee. (2019). Global and regional diabetes prevalence estimates for 2019 and projections for

2030 and 2045: Results from the International Diabetes Federation Diabetes Atlas. Diabetes research and clinical practice, 157, 107843.

[3] Rubin, D. J. (2015). Hospital readmission of patients with diabetes. Current diabetes reports, 15(4), 17.

[4] Rubin, D. J., Donnell-Jackson, K., Jhingan, R., Golden, S. H., & Paranjape, A. (2014). Early readmission among patients with diabetes: a qualitative assessment of contributing factors. Journal of Diabetes and its Complications, 28(6), 869-873.

[5] Cui, S., Wang, D., Wang, Y., Yu, P. W., & Jin, Y. (2018). An improved support vector machine-based diabetic readmission prediction. Computer methods and programs in biomedicine, 166, 123-135.

[6] Hammoudeh, A., Al-Naymat, G., Ghannam, I., & Obied, N. (2018). Predicting hospital readmission among diabetics using deep learning. Procedia Computer Science, 141, 484-489.

[7] Rubin, D. J., Handorf, E. A., Golden, S. H., Nelson, D. B., McDonnell, M. E., & Zhao, H. (2016). Development and validation of a novel tool to predict hospital readmission risk among patients with diabetes. Endocrine Practice, 22(10), 1204-1215.

[8] Bhuvan, M. S., Kumar, A., Zafar, A., & Kishore, V. (2016). Identifying diabetic patients with high risk of readmission. arXiv preprint arXiv:1602.04257.

[9] Duggal, R., Shukla, S., Chandra, S., Shukla, B., & Khatri, S. K. (2016). Impact of selected pre-processing techniques on prediction of risk of early readmission for diabetic patients in India. International Journal of Diabetes in Developing Countries, 36(4), 469-476. [10] WHO, Global report of 2004, [link]

[10] Sharma, A., Agrawal, P., Madaan, V., & Goyal, S. (2019, June). Prediction on diabetes patient's hospital readmission rates. In Proceedings of the Third International Conference on Advanced Informatics for Computing Research (pp. 1-5).