Diabetes Patient's Hospital Readmission

INFO 6105-18687

Anish Sridhar
Anuja Thawali
Nikhil Prabhu Shankar
Sai Swaroop Reddy Pothireddy

Introduction

A hospital readmission is when a patient who is discharged from the hospital, gets re-admitted again within a certain period. Hospital readmission rates for certain conditions are now considered an indicator of hospital quality, and affect the cost of care adversely. For this reason, the Centers for Medicare & Medicaid Services established the Hospital Readmissions Reduction Program which aims to improve the quality of care for patients and reduce health care spending by applying payment penalties to hospitals that have more than expected readmission rates for certain conditions. Although diabetes is not yet included in the penalty measures, the program is regularly adding new disease conditions to the list, now totaling 6 for FY2018. In 2011, American hospitals spent over $41 billion on diabetic patients who got readmitted within 30 days of discharge. Being able to determine the factors that lead to higher readmission in such patients, and correspondingly being able to predict which patients will get readmitted can help hospitals save millions of dollars while improving quality of care.

Keywords

- Diabetes, Machine Learning, Data Science
- Dummy variables, correlation matrix
- Logistic Regression, Random Forest, K-nearest neighbours

Objectives

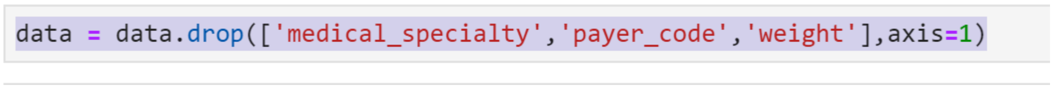
The goal can broadly be differentiated into 4 parts:

- Help understanding the major parameters causing diabetes
- Clean the dataset to get appropriate dataset for fast calculations
- Train the model to predict what factors are the strongest predictors of hospital readmission in diabetic patients and predict effective treatments for diabetes in turn reducing the readmission into the hospital
- Approaches to improve the accuracy of prediction using medical data with various machine learning algorithms and methods

Methodology

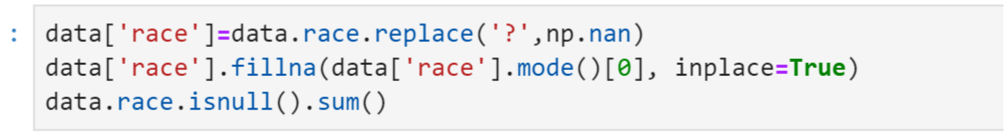
1. Dataset cleaning

   We modified the dataset to drop columns ‘medical_speciality’, ‘payer_code’,’weight’ as they seemed irrelevant to calculate readmission of a diabetic patient.



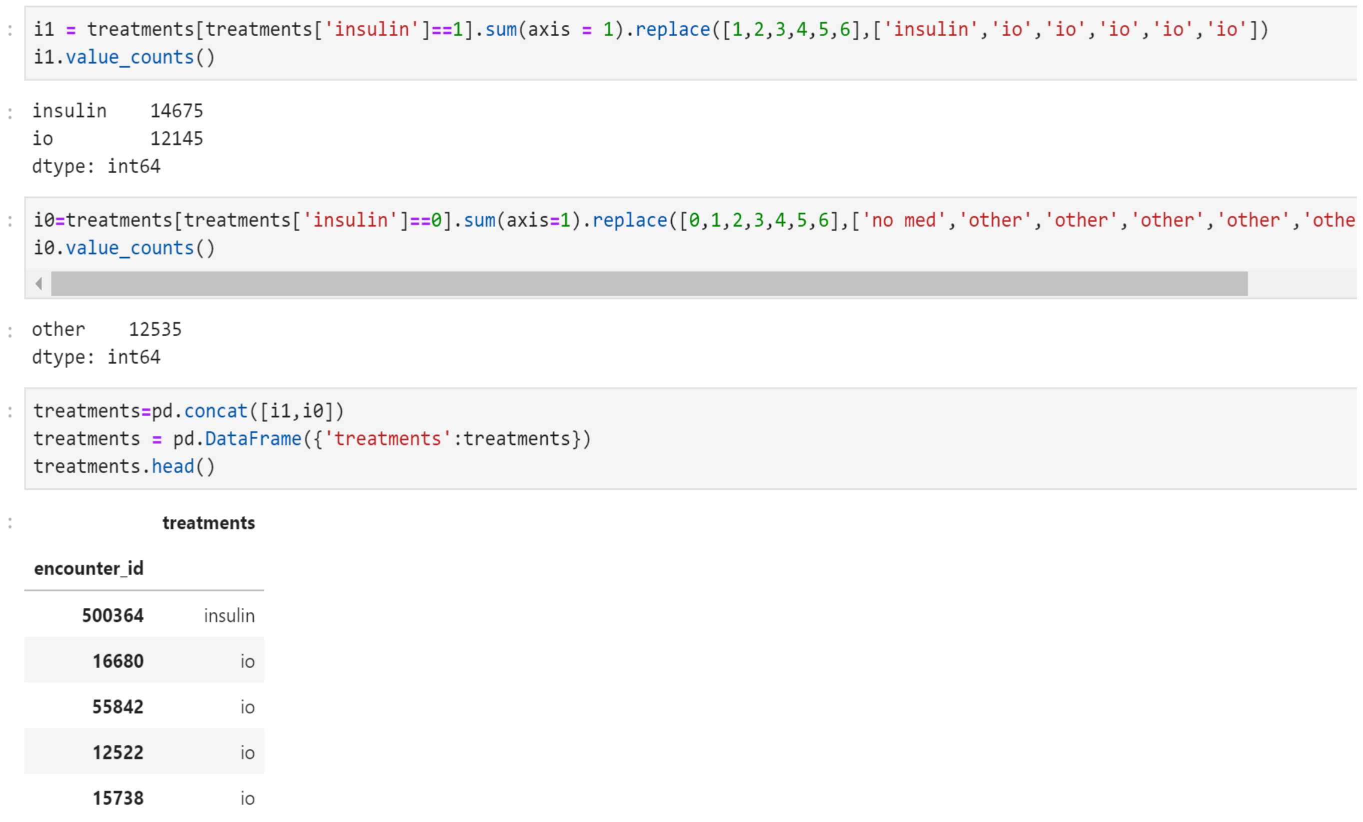
```
data = data.drop(['medical_specialty','payer_code','weight'],axis=1)
```

   Column ‘Race’ had some missing numbers which were filled with a question mark. We replaced it with a nan representation.



```
: data['race']=data.race.replace('?',np.nan)
  data['race'].fillna(data['race'].mode()[0], inplace=True)
  data.race.isnull().sum()
```

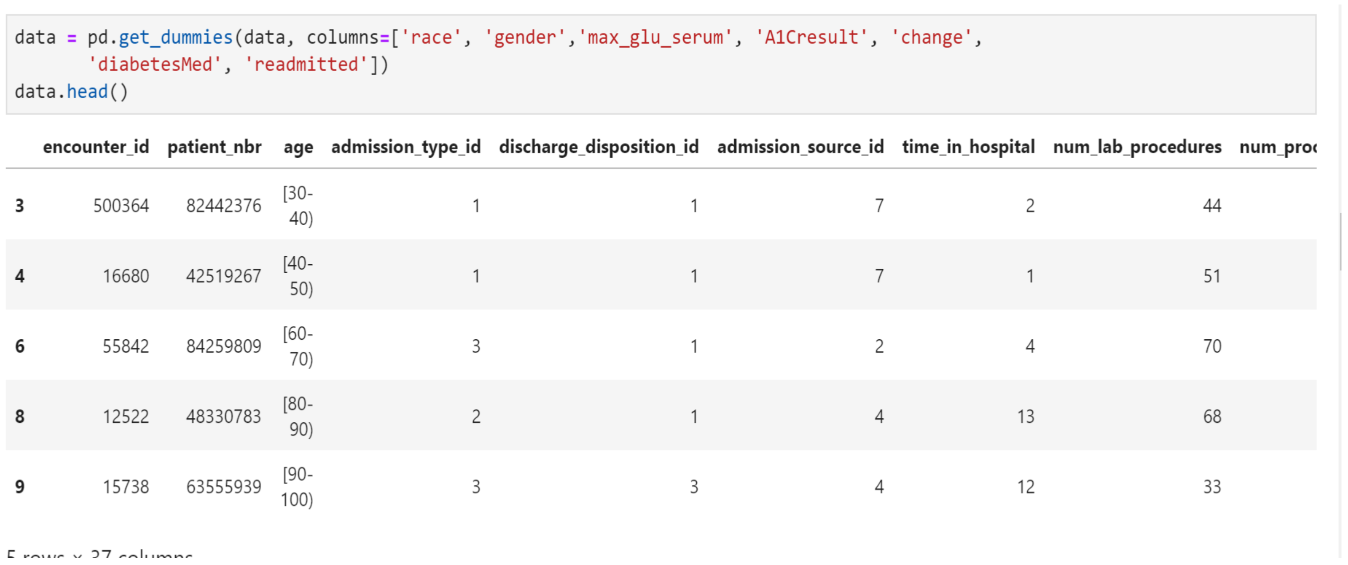
2. Implement feature engineering for following:
   a. Custom encode medicine data for simplicity
   b. Count total medicines prescribed to patient
   c. Introduce a new column ‘Treatment’ to identify prescription of insulin, other drugs or both for quick identification



```
: i1 = treatments[treatments['insulin']==1].sum(axis = 1).replace([1,2,3,4,5,6],['insulin','io','io','io','io','io'])
  i1.value_counts()

: insulin    14675
  io         12145
  dtype: int64

: i0=treatments[treatments['insulin']==0].sum(axis=1).replace([0,1,2,3,4,5,6],['no med','other','other','other','other','othe
  i0.value_counts()

: other    12535
  dtype: int64

: treatments=pd.concat([i1,i0])
  treatments = pd.DataFrame({'treatments':treatments})
  treatments.head()

:                treatments
  encounter_id
        500364      insulin
         16680           io
         55842           io
         12522           io
         15738           io
```

d. Used dummy variables feature to split some columns in various columns to get a detailed insight.

   To convert our categorical features to numbers, we will use a technique called one-hot encoding. In one-hot encoding, we create a new column for each unique value in that column. Then the value of the column is 1 if the sample has that unique value or 0 otherwise. To create these one-hot encoding columns, we used the get_dummies function provided by pandas for columns 'race', ‘gender', 'max_glu_serum', 'A1Cresult', 'change', 'diabetesMed', 'readmitted'.



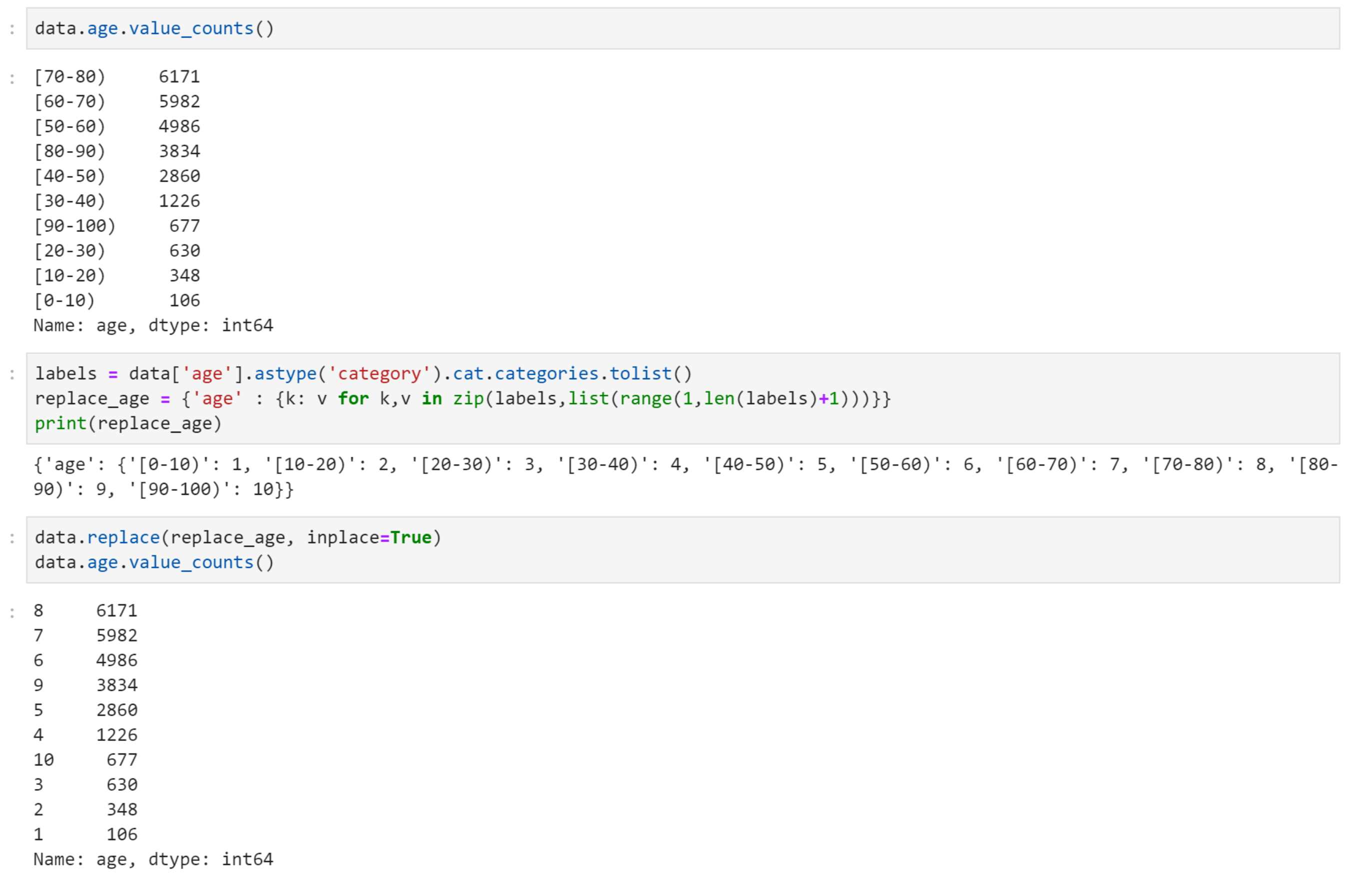
```
data = pd.get_dummies(data, columns=['race', 'gender','max_glu_serum', 'A1Cresult', 'change',
        'diabetesMed', 'readmitted'])
data.head()
```

	encounter_id	patient_nbr	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	num_proc
3	500364	82442376	[30-40)	1	1	7	2	44	
4	16680	42519267	[40-50)	1	1	7	1	51	
6	55842	84259809	[60-70)	3	1	2	4	70	
8	12522	48330783	[80-90)	2	1	4	13	68	
9	15738	63555939	[90-100)	3	3	4	12	33	

5 rows × 37 columns

e. Modified columns that had a range in the value with a finite value for easy calculations further

Our dataset has column of age that was represented as a range for age of a person. We have replaced the range value with a finite value for each range for easy calculations further.



```
data.age.value_counts()
```

```
[70-80)     6171
[60-70)     5982
[50-60)     4986
[80-90)     3834
[40-50)     2860
[30-40)     1226
[90-100)     677
[20-30)      630
[10-20)      348
[0-10)       106
Name: age, dtype: int64
```

```
labels = data['age'].astype('category').cat.categories.tolist()
replace_age = {'age' : {k: v for k,v in zip(labels,list(range(1,len(labels)+1)))}}
print(replace_age)
```

```
{'age': {'[0-10)': 1, '[10-20)': 2, '[20-30)': 3, '[30-40)': 4, '[40-50)': 5, '[50-60)': 6, '[60-70)': 7, '[70-80)': 8, '[80-
90)': 9, '[90-100)': 10}}
```

```
data.replace(replace_age, inplace=True)
data.age.value_counts()
```

```
8     6171
7     5982
6     4986
9     3834
5     2860
4     1226
10     677
3      630
2      348
1      106
Name: age, dtype: int64
```

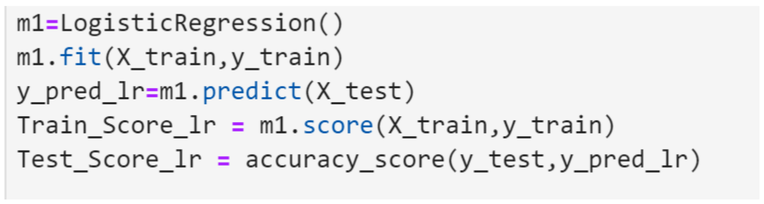
3. Implement feature identification with a co-relation matrix to get relation between various features

4. Implement predictive modelling

   The data cleaning and feature engineering of dataset help us to get a more precise dataset with relevant columns and their values

   a. Logistic Regression

      Logistic regression is an excellent model to use when the features are linearly separable. One advantage of logistic regression is the model is interpretable — i.e. we know which features are important for predicting positive or negative. We can fit logistic regression using the following code from scikit-learn.



```
m1=LogisticRegression()
m1.fit(X_train,y_train)
y_pred_lr=m1.predict(X_test)
Train_Score_lr = m1.score(X_train,y_train)
Test_Score_lr = accuracy_score(y_test,y_pred_lr)
```

   b. KNN

      KNN is one the simplest machine learning models. For a given sample point, the model looks at the k closest datapoints and determines the probability by counting the number of positive labels divided by K. We found this model to be easy to implement and understand but comes at the disadvantage of being sensitivity to K and takes a long time to evaluate if the number of trained samples is large.

```
m2 = KNeighborsClassifier()
m2.fit(X_train,y_train)
y_pred_knn = m2.predict(X_test)
Train_Score_knn = m2.score(X_train,y_train)
Test_Score_knn = accuracy_score(y_test,y_pred_knn)
```

c. Bernouli Naive Bayes

Naive Bayes is another model occasionally used in machine learning. In Naive Bayes, we utilize Bayes Rule to calculate the probabilities. The "naive" part of this model is that it assumes all the features are independent (which is generally not the case). This works well for natural language processing models, but let's try it out here anyways. Bernouli Naive Bayes is better while working with binary features.

```
m3=BernoulliNB()
m3.fit(X_train,y_train)
y_pred_bnb=m3.predict(X_test)
Train_Score_bnb = m3.score(X_train,y_train)
Test_Score_bnb = accuracy_score(y_test,y_pred_bnb)
```

d. Decision Tree

The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data).

The machine learning behind this method is to figure out which variable and which threshold to use at every split. One advantage of tree-based methods is that they have no assumptions about the structure of the data and are able to pick up non-linear effects if given sufficient tree depth. We can fit decision trees using the following code

```
m4 = DecisionTreeClassifier()
m4.fit(X_train,y_train)
y_pred_dt=m4.predict(X_test)
Train_Score_dt = m4.score(X_train,y_train)
Test_Score_dt = accuracy_score(y_test,y_pred_dt)
```

e. Random Forest

One disadvantage of decision trees is that they tend overfit very easily by memorizing the training data. As a result, random forests were created to reduce the overfitting. In random forest models, multiple trees are created and the results are aggregated. The trees in a forest are decorrelated by using a random set of samples and random number of features in each tree. In most cases, random forests work better than decision trees because they are able to generalize more easily.

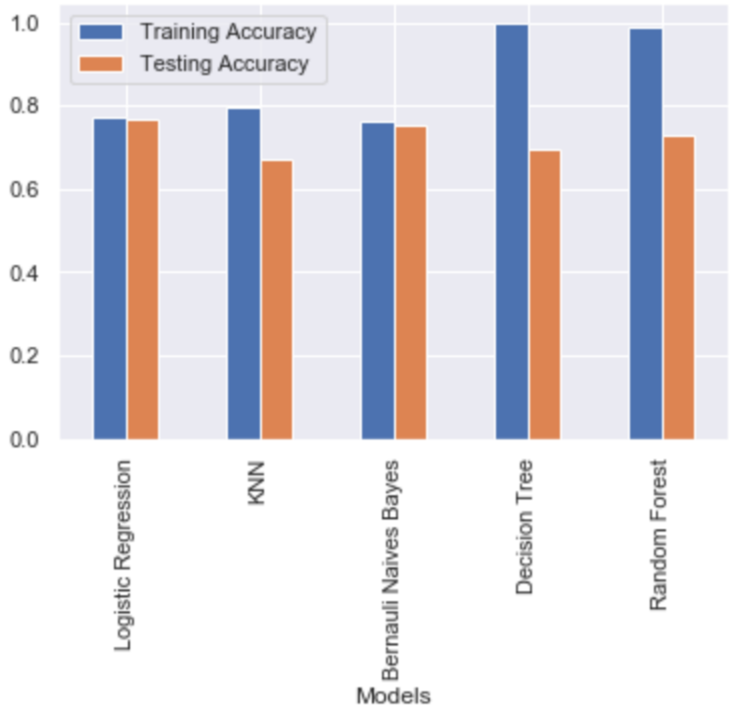
```
m5 = RandomForestClassifier()
m5.fit(X_train,y_train)
y_pred_rf=m5.predict(X_test)
Train_Score_rf = m5.score(X_train,y_train)
Test_Score_rf = accuracy_score(y_test,y_pred_rf)
```

5. Analyze accuracy for each model

Result Analysis

Accuracy prediction:

The following graph helps us to compare the accuracies of various models applied for getting the records of readmitted patients and get the best visualization of the result for better understanding



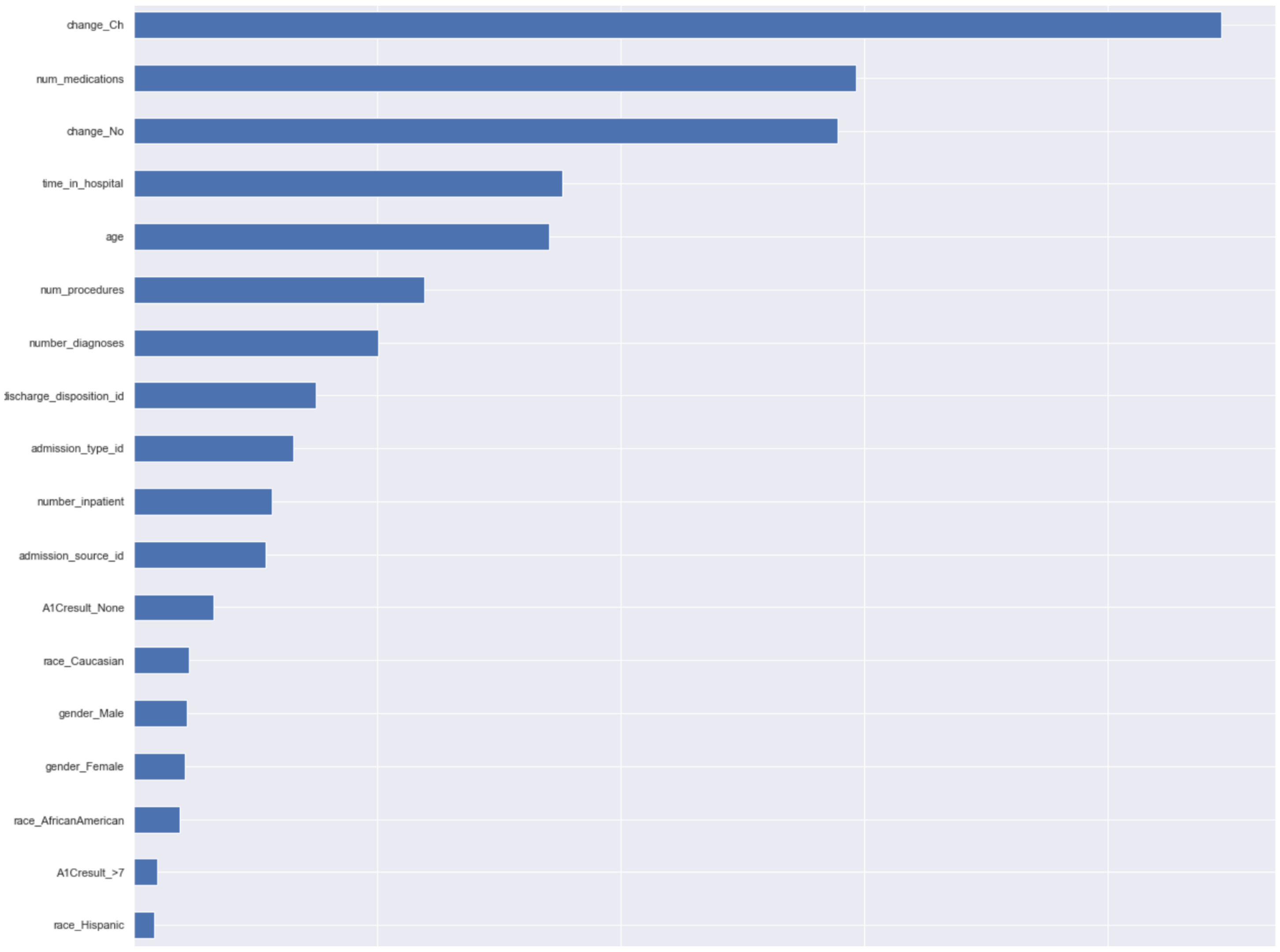
We were able to get an accuracy of nearly 76 % with the help of Logistic Regression, which is very good in real time analysis to verify our accuracy. Below is classification report for Logistic regression to show the precisions, recalls and f1-score for more details:

```
              precision    recall  f1-score   support

     insulin       0.94      0.61      0.74      4375
          io       0.67      0.96      0.79      3671

    accuracy                           0.77      8046
   macro avg       0.81      0.78      0.76      8046
weighted avg       0.82      0.77      0.76      8046
```

This table gives us the weighted average for precision to 82% which verifies our model accuracy to be realistic

Feature importance:

The graph below shows us important features of the data set that help us identify readmitted patients easily. Features like number of medicines and change in medication are important factors.



Dataset and Specifications

https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008

The data set represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 50 features representing patient and hospital outcomes. Information was extracted from the database for encounters that satisfied the following criteria:

1. It is an inpatient encounter (a hospital admission).
2. It is a diabetic encounter, that is, one during which any kind of diabetes was entered to the system as a diagnosis.
3. The length of stay was at least 1 day and at most 14 days.
4. Laboratory tests were performed during the encounter.
5. Medications were administered during the encounter.

In this database, we have 3 different output:

1. No readmission;
2. A readmission in less than 30 days (this situation is not good, because maybe your treatment was not appropriate);
3. A readmission in more than 30 days (this one is not so good as well the last one, however, the reason can be the state of the patient.

The data contains such attributes as patient number, gender, age, admission type, time in hospital, medical specialty of admitting physician, number of lab tests performed, HbA1c test result, diagnosis, number of medications, diabetic medications, number of outpatient, inpatient, and emergency visits in the year before the hospitalization, etc.
Contains one csv file:
➔ Diabetes dataset: 102K records with 50 columns

Resources

1) https://towardsdatascience.com/machine-learning-for-diabetes-562dd7df4d42

2) https://lightgbm.readthedocs.io/en/latest/Python-API.html

3) https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

4) https://scikit-learn.org/stable/model_selection.html#model-selection

Project Deliverables

Following is a complete list of all project deliverables:

- Python file
- PowerPoint presentation
- Project report
- Dataset

Conclusion

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 model was Logistic regression classifier. The model was able to predict 76% of the treatments and helped to reduce readmission of patients.
We have also found that Age, number of medications, change in medication and time in the hospital are major factors that cause readmission.