

Group 6 Project

Diabetic Patients Data

Final Report

Shruti Kamble, Nilay Anand, Ganeeth Reddy Atmakuri

College of Prof. Studies, Analytics



ALY 6110: Data Management and Big Data

Prof. Daya Rudhramoorthi

Introduction

For our final project, we have chosen to analyze and model diabetic patient data. The dataset contains information about the patient's age, gender, and race with almost 100,000 entries of patient records. The dataset can be viewed [here](#). It also has diagnoses received in tests, medication administered and change in diagnosis of the patient. Finally, it has data about time spent at a hospital, number of procedures and various other hospital related metrics but we will not be using it for our current analysis. The goal of this analysis is to see if age, gender, and race are factors affecting diagnosis and what effect different medication (primarily insulin and Diabetes medicine) has on the patient. Relevant graphs for analysis have been included. We have included a Tableau dashboard for information.

For discovering useful information, we are using various methodologies to reach a conclusion. This will be done through a process of importing, cleaning, inspecting, transforming, and modeling data using statistical and analytical tools. Further, analysis will involve regression analysis, descriptive analysis, predictive analysis along with effective visualization. Regression analysis will help in evaluating the relationship between sets of variables.

We learnt about the data beyond basic analysis is

- This database has a wealth of information. Because several categorical features have a large number of possible values, those variables will need to be changed (especially the diag 1, diag 2, and diag 3 variables, which have over 700 different values).
- Some traits, particularly those related to medications, are not evenly distributed (a lot of No).

- We're going to eliminate the weight, _payer code_, and _medical specialization_ features because they have a lot of missing information.
- There are 101766 possible values for _encounterid, but only 71518 different values for _patientnbr: Some patients returned multiple times (which is common because we track readmissions), and we can recognize them.

Analysis

1. Through the analysis, we will answer the following question:

- In diabetic patients, what factors are the strongest predictors of hospital readmission?
- How well can we predict hospital readmission in this dataset with limited features?

2. Exploratory Data Analysis

We have used,

- Excel for initial analysis; to get an idea of entities in the data
- Tableau for visualization of data before processing, and comparing in different scenarios
- Pyspark for tasks like loading, treating missing values, and building model
- Pandas for creating data frame that will be used in modeling
- Sklearn python library for building predictive model

Starting with excel, we've viewed all the columns in the data frame and observed that some columns had data that is relevant to the business problem.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	encounter	patient_id	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	payer_code	medical_specialty	num_lab_procedures	num_procedures	num_medications	number_of_visits	number_of_encounters
1	2278392	8222157	Caucasian	Female	[0-10]	?	6	25	1	1	?	Pediatrics	41	0	1	0	0
2	149190	55629189	Caucasian	Female	[10-20]	?	1	1	7	3	?	?	59	0	18	0	0
3	64410	86047875	AfricanAm	Female	[20-30]	?	1	1	7	2	?	?	11	5	13	2	0
4	500364	82442376	Caucasian	Male	[30-40]	?	1	1	7	2	?	?	44	1	16	0	0
5	16680	42519267	Caucasian	Male	[40-50]	?	1	1	7	1	?	?	51	0	8	0	0
6	35754	82637451	Caucasian	Male	[50-60]	?	2	1	2	3	?	?	31	6	16	0	0
7	55842	84259809	Caucasian	Male	[60-70]	?	3	1	2	4	?	?	70	1	21	0	0
8	63768	1.15E+08	Caucasian	Male	[70-80]	?	1	1	7	5	?	?	73	0	12	0	0
9	12522	48330783	Caucasian	Female	[80-90]	?	2	1	4	13	?	?	68	2	28	0	0
10	15738	63555939	Caucasian	Female	[90-100]	?	3	3	4	12	?	InternalMed	33	3	18	0	0
11	28236	89869032	AfricanAm	Female	[40-50]	?	1	1	7	9	?	?	47	2	17	0	0
12	36900	77391171	AfricanAm	Male	[60-70]	?	2	1	4	7	?	?	62	0	11	0	0
13	40926	85504905	Caucasian	Female	[40-50]	?	1	3	7	7	?	Family/Gen	60	0	15	0	1
14	42570	77586282	Caucasian	Male	[80-90]	?	1	6	7	10	?	Family/Gen	55	1	31	0	0
15	62256	49726791	AfricanAm	Female	[60-70]	?	3	1	2	1	?	?	49	5	2	0	0
16	73578	86328819	AfricanAm	Male	[60-70]	?	1	3	7	12	?	?	75	5	13	0	0
17	77076	92519352	AfricanAm	Male	[50-60]	?	1	1	7	4	?	?	45	4	17	0	0
18	84222	1.09E+08	Caucasian	Female	[50-60]	?	1	1	7	3	?	Cardiology	29	0	11	0	0
19	89682	1.07E+08	AfricanAm	Male	[70-80]	?	1	1	7	5	?	?	35	5	23	0	0
20	148530	69422211	?	Male	[70-80]	?	3	6	2	6	?	?	42	2	23	0	0
21	150006	22864131	?	Female	[50-60]	?	2	1	4	2	?	?	66	1	19	0	0
22	150048	21239181	?	Male	[60-70]	?	2	1	4	2	?	?	36	2	11	0	0
23	182796	63000108	AfricanAm	Female	[70-80]	?	2	1	4	2	?	?	47	0	12	0	0
24	183930	1.07E+08	Caucasian	Female	[80-90]	?	2	6	1	11	?	?	42	2	19	0	0
25	216156	62718876	AfricanAm	Female	[70-80]	?	3	1	2	3	?	?	19	4	18	0	0
26	221634	21861756	Other	Female	[50-60]	?	1	1	7	1	?	?	33	0	7	0	0
27	236316	40523301	Caucasian	Male	[80-90]	?	1	3	7	6	?	Cardiology	64	3	18	0	0

Figure 1: Excel Sheet

As seen in Figure 1 Columns like weight, payer_code, medical_speciality had a lot of missing values and were irrelevant to the problem. These columns were not considered for initial analysis and were removed in the later stages.

Next, Using the power of Tableau, we built a comprehensive dashboard to look at the data in different scenarios. We look at diagnosis of patients by age, gender and race. Also, we save if the patient were readmitted or there was a change in patient condition given certain sets of treatment.

The Patient Dashboard can be viewed here:

https://public.tableau.com/app/profile/nalay.anand/viz/ALY6110_Group06_Dashboard/PatientDiagnosis?publish=yes



Figure 2: Tableau Dashboard

As seen in Figure 2, filters by age, race, gender and different types of medication administered were added to the graph for better visualization of scenarios.

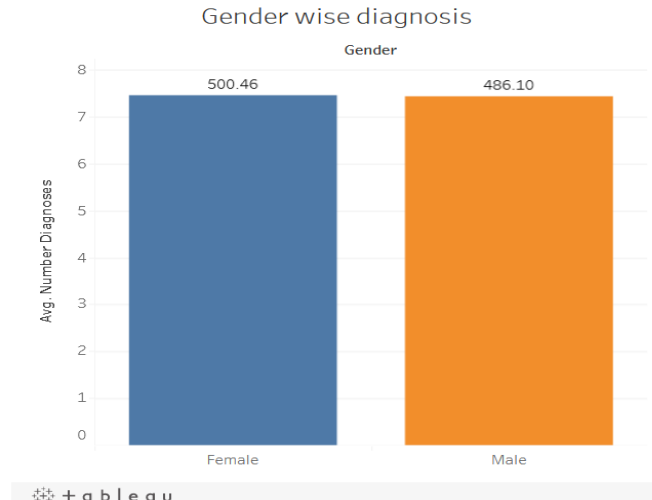


Figure 3: Gender wise diagnosis

Focusing on diagnoses by gender: Taking the average number of diabetes diagnoses for each gender and using the average diagnosis as the label, displays the graph shown in figure 3.

The number of diagnoses seems equal in case of male and female but the average diagnosis value is higher in females compared to males.

So, gender has some correlation with the intensity of the disease.

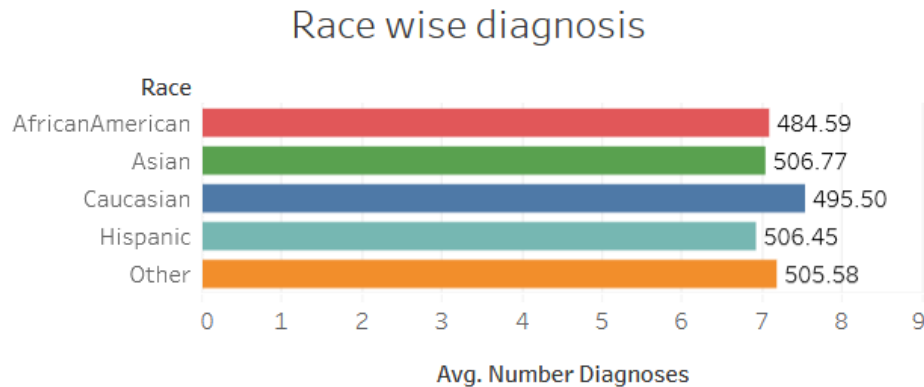


Figure 4: Race wise diagnosis

Focusing on diagnoses by race: Plotting the average amount of diagnosis for each race and adding the average diagnosis score as labels, displays the graph shown in Figure 4. There is not much difference between the races in terms of average diabetic score but the average number of cases is much higher for the Caucasian race. As per the stats, there is Caucasians in 73 percent of all our data. And the other 22 percent is divided into African Americans, Hispanics, Asians and Others.

Most of the patients are Caucasian, followed by African Americans. Although the Other values are fewer than Caucasian, we see that the Readmitted Probability is almost close to Caucasian.

Possible reasons for this will be discussed further.

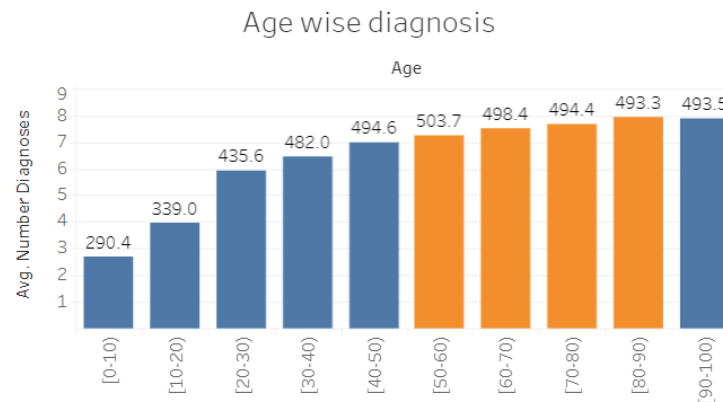


Figure 5: Age wise diagnosis

Focusing on diagnoses by age: Plotting the average of diagnoses for each age group and adding labels as average diagnosis score, displays the graph shown in Figure 5. In this case there is a clear formation of clusters. One for the age groups of 50 to 90 and second in the age group of 0-50 and 90 – 100. The number of diagnoses increase with age and the severity of the disease gets much higher. So, age has a very apparent effect on diabetes and its intensity. Possible reasons for this will be discussed further.

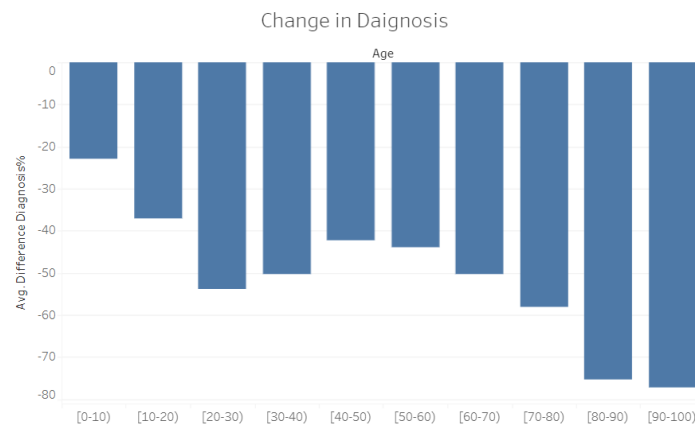


Figure 6: Percentage change in diagnosis by age

Figure 6 shows the patients in which a change was recorded after medication. The columns bar represent difference between their second diagnosis and first diagnosis, calculated by the following formula:

$$\text{Change Diag\%} = \frac{(\text{Diag1} - \text{Daig2})}{\text{Daig1}} \times 100$$

The graph shows that older individuals in the range of 20 and above are more affected by medication. Whereas, children having lower diagnosis have lower change in the diagnosis.

Next, effectiveness of treatment was checked for multiple scenarios. Our analysis was focused on insulin and diabetes meds as these are the key treatments given to the patient with diabetes. The analysis gave us the following plots.

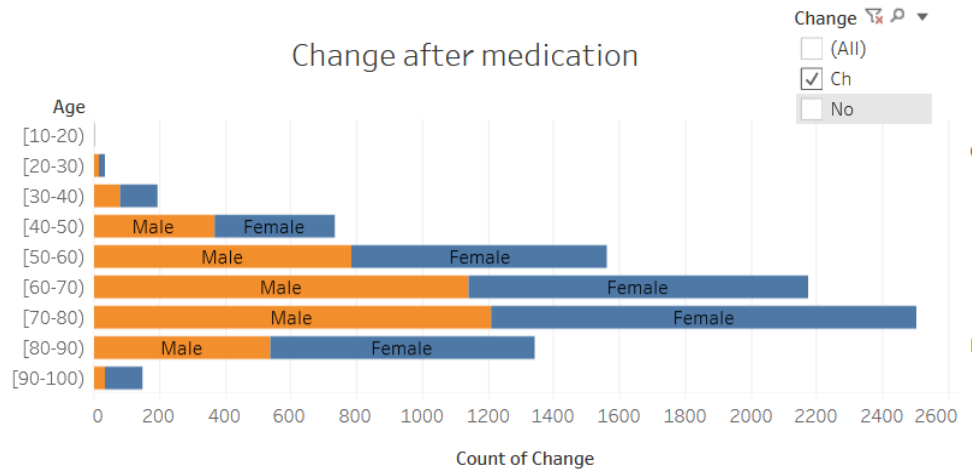


Figure 7: Change in patient given only Diabetes med

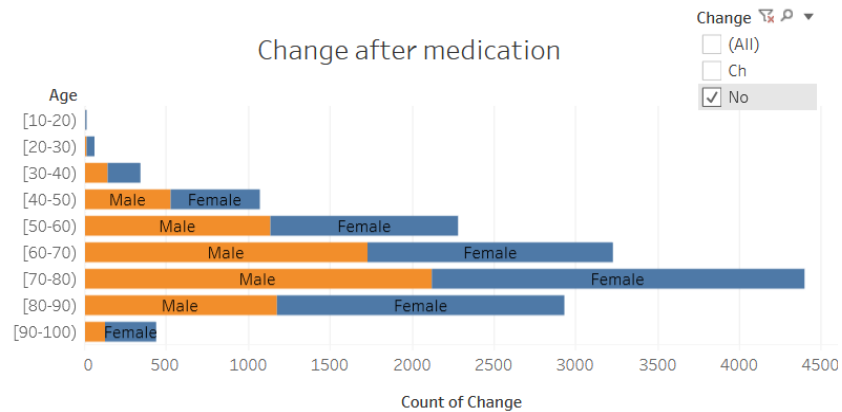


Figure 8: No Change in patient given only Diabetes meds

Checking for patients that received the Diabetes med, as we see in Figure 6 and Figure 7. It can be concluded that Diabetes meds are not highly effective as the number of cases in which no change was seen is much higher in each age group. Although, there were no cases in which there was a change if the diabetes medication was not given. So, it is better to get the treatment and the effect can be analyzed on a case by case basis.

Next, we checked if patients were readmitted or not in scenarios where diabetes medication was given.

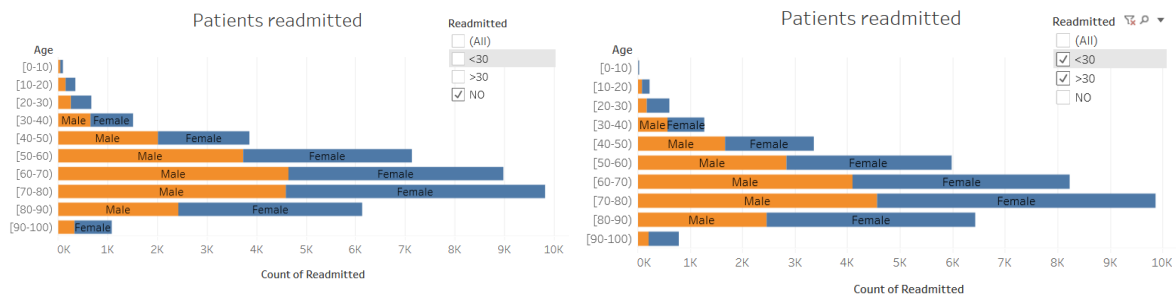


Figure 9: Diabetes med administered

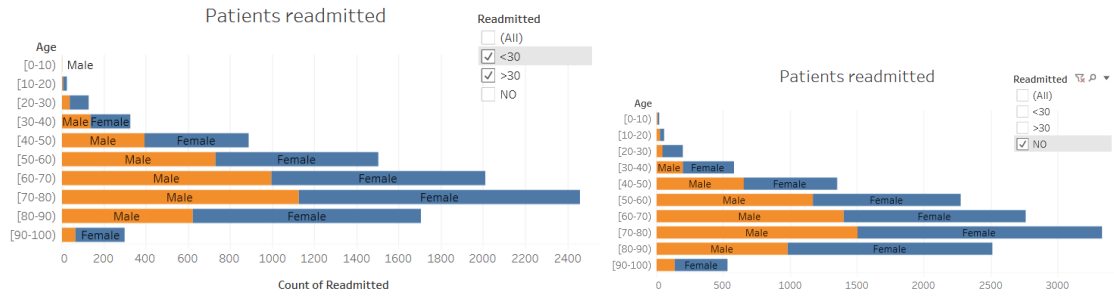


Figure 10: Diabetes med not administered

As seen in Figure 8 and Figure 9, in cases where Diabetes med was given as part of treatment, less number of patients were readmitted to the hospital. The overall cost of treatment for such cases would be lower and less time was spent in the hospital.

When Diabetes med was not given as part of the treatment, the readmission rate is much higher and puts patients at much higher risk.

3. Data Preparation

For cleaning the data the dataset was downloaded using terminal command within a python notebook, extracted and loaded as a spark data frame.

```

conf = pyspark.SparkConf()
conf.setAppName('mySparkApp')
conf.setMaster('local')
sc = pyspark.SparkContext(conf=conf)
spark = SparkSession(sc)

[4] wget -q http://archive.ics.uci.edu/ml/machine-learning-databases/00296/dataset_diabetes.zip

[5] unzip dataset_diabetes.zip

Archive: dataset_diabetes.zip
  inflating: dataset_diabetes/diabetic_data.csv
  inflating: dataset_diabetes/IDs_mapping.csv

[6] diabetes_df = spark.read.option('header','true').csv('/content/dataset_diabetes/diabetic_data.csv',inferSchema=True)

[7] diabetes_df.show()

```

encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	payer_code	medical_specialty	num_lab
2278392	8222157	Caucasian	female	[0-10]	?	6	25	1	1	?	Pediatrics-Endocr...	?
149190	55629189	Caucasian	female	[10-20]	?	1	1	?	3	?		?
64410	86047875	AfricanAmerican	female	[20-30]	?	1	1	?	2	?		?
500364	82442376	Caucasian	Male	[30-40]	?	1	1	?	2	?		?
16680	42519267	Caucasian	Male	[40-50]	?	1	1	?	1	?		?
35754	82637451	Caucasian	Male	[50-60]	?	2	1	?	3	?		?
55842	84259809	Caucasian	Male	[60-70]	?	3	1	?	4	?		?

Figure 11: Loading data into spark

On viewing the data using the show command, we see that multiple columns have missing values. These columns were dropped from the data frame.

```

[11] diabetes_df = diabetes_df.drop('weight','payer_code','medical_specialty')

[12] meds = diabetes_df.columns[21:-3]

for i in meds:
    percent_No = (diabetes_df.filter(diabetes_df[i] == "No").groupby().count().first()[0]/diabetes_df.groupby().count().first()[0])*100
    if percent_No > 99.0:
        print(i, percent_No)
        diabetes_df = diabetes_df.drop(i)

nateglinide 99.30919953619087
chlorpropamide 99.91549240414284
acetohexamide 99.99901735353654
tolbutamide 99.97739913134053
acarbose 99.69734488925575
miglitol 99.9626594343887
troglitazone 99.99705206060963
tolazamide 99.96167678792524
examide 100.0
citoglipton 100.0
glyburide-metformin 99.3062515968005
glipizide-metformin 99.98722559597508
glimepiride-pioglitazone 99.99901735353654
metformin-rosiglitazone 99.99803470707309
metformin-pioglitazone 99.99901735353654

```

Figure 12: Dropping data

As seen in Figure 11, we also checked the percentage of cases in which a medication was not being administered and if the percentage was greater than 99%, that column was dropped as this medication was not relevant to the diagnosis of diabetes in the patient.

We opted to eliminate weight variables since they include around 98 percent of the missing values. Filling those missing values has little relevance, thus we decided to drop them. We also removed the variables Payer code and medical speciality because they had approximately 40% missing information. Race, diag 1, diag 2, diag 3, and gender variables have a lot less missing values than the other qualities we discarded, therefore we decided to drop them as well.

After dropping the irrelevant data, we checked the schema of the data to view what columns were left and what was the data type in each column.

```
[14] diabetes_df.printSchema()

root
|-- encounter_id: integer (nullable = true)
|-- patient_nbr: integer (nullable = true)
|-- race: string (nullable = true)
|-- gender: string (nullable = true)
|-- age: string (nullable = true)
|-- admission_type_id: integer (nullable = true)
|-- discharge_disposition_id: integer (nullable = true)
|-- admission_source_id: integer (nullable = true)
|-- time_in_hospital: integer (nullable = true)
|-- num_lab_procedures: integer (nullable = true)
|-- num_procedures: integer (nullable = true)
|-- num_medications: integer (nullable = true)
|-- number_outpatient: integer (nullable = true)
|-- number_emergency: integer (nullable = true)
|-- number_inpatient: integer (nullable = true)
|-- diag_1: string (nullable = true)
|-- diag_2: string (nullable = true)
|-- diag_3: string (nullable = true)
|-- number_diagnoses: integer (nullable = true)
|-- max_glu_serum: string (nullable = true)
|-- A1cresult: string (nullable = true)
|-- metformin: string (nullable = true)
|-- repaglinide: string (nullable = true)
|-- glimepiride: string (nullable = true)
|-- glipizide: string (nullable = true)
|-- glyburide: string (nullable = true)
|-- pioglitazone: string (nullable = true)
|-- rosiglitazone: string (nullable = true)
|-- insulin: string (nullable = true)
|-- change: string (nullable = true)
|-- diabetesMed: string (nullable = true)
|-- readmitted: string (nullable = true)
```

Figure 13: Final table schema

While working with the data, a point of concern was that multiple columns had “?” symbols in place where data was not available. These symbols needed to be replaced so that data can easily be formatted for modeling. This was done as shown in Figure 13.

```

diabetes_df_2 = diabetes_df
from pyspark.sql.functions import when
from pyspark.sql.types import FloatType

diabetes_df_2 = diabetes_df_2.withColumn("race", \
    when(diabetes_df_2["race"] == '?', None).otherwise(diabetes_df_2["race"]))
diabetes_df_2 = diabetes_df_2.withColumn("diag_1", \
    when(diabetes_df_2["diag_1"] == '?', None).otherwise(diabetes_df_2["diag_1"]))
diabetes_df_2 = diabetes_df_2.withColumn("diag_2", \
    when(diabetes_df_2["diag_2"] == '?', None).otherwise(diabetes_df_2["diag_2"]))
diabetes_df_2 = diabetes_df_2.withColumn("diag_3", \
    when(diabetes_df_2["diag_3"] == '?', None).otherwise(diabetes_df_2["diag_3"]))
diabetes_df_2 = diabetes_df_2.withColumn("diag_1", diabetes_df_2["diag_1"].cast(FloatType()))
diabetes_df_2 = diabetes_df_2.withColumn("diag_2", diabetes_df_2["diag_2"].cast(FloatType()))
diabetes_df_2 = diabetes_df_2.withColumn("diag_3", diabetes_df_2["diag_3"].cast(FloatType()))
diabetes_df_2.show()

```

encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	num_procedures	num_medications
2278392	8222157	Caucasian	Female	[0-10]	6	25	1	1	41	0	
149198	55629189	Caucasian	Female	[10-20]	1	1	7	3	59	0	
64418	80847875	AfricanAmerican	Female	[20-30]	1	1	7	2	11	5	
508364	82442376	Caucasian	Male	[30-40]	1	1	7	2	44	1	
16688	42519267	Caucasian	Male	[40-50]	1	1	7	1	51	0	
35754	82637451	Caucasian	Male	[50-60]	2	1	2	3	31	6	
55842	84259889	Caucasian	Male	[60-70]	3	1	2	4	70	1	
63768	114882964	Caucasian	Male	[70-80]	1	1	7	5	73	0	
12522	48330783	Caucasian	Female	[80-90]	2	1	4	13	68	2	
15738	63555939	Caucasian	Female	[90-100]	3	1	4	12	33	3	
28236	89869032	AfricanAmerican	Female	[40-50]	1	1	7	9	47	0	
36988	77391171	AfricanAmerican	Male	[60-70]	2	1	4	7	62	0	
40926	85504905	Caucasian	Female	[40-50]	1	3	7	7	68	0	
42578	77586282	Caucasian	Male	[80-90]	1	6	7	10	55	1	
62256	49726791	AfricanAmerican	Female	[60-70]	3	1	2	1	49	5	
73578	86348819	AfricanAmerican	Male	[60-70]	1	3	7	12	75	5	
77676	92519352	AfricanAmerican	Male	[50-60]	1	1	7	4	45	4	
84222	108662661	Caucasian	Female	[50-60]	1	1	7	3	29	0	
89682	107389323	AfricanAmerican	Male	[70-80]	1	1	7	5	35	5	
148538	69422211	null	Male	[70-80]	3	1	2	6	42	2	

only showing top 20 rows

7s completed at 6:57 PM

Figure 14: Replacing missing values with Null

The dataset contains some data in the form of objects and it requires encoding in numerical form. Also, our goal is to predict whether a patient will be readmitted or not. The columns contain three values and these need to be converted to binary yes and no.

```

[ ] diabetes_pd_df['readmitted'].unique()

array(['NO', '>30', '<30'], dtype=object)

[ ] diabetes_pd_df['readmitted'] = diabetes_pd_df['readmitted'].replace({'>30':'YES', '<30':'YES'})

```

Figure 15: Convert readmitted in form of yes and no

The data has a few columns which contain data in object form and using label encoder all the columns were converted to integers.

```

from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
for i in diabetes_pd_df.columns:
    if diabetes_pd_df[i].dtype == 'object':
        diabetes_pd_df[i] = labelencoder.fit_transform(diabetes_pd_df[i])
    print(i, diabetes_pd_df[i].dtype)

race int64
gender int64
age int64
max_glu_serum int64
A1cresult int64
metformin int64
repaglinide int64
glimepiride int64
glipizide int64
glyburide int64
pioglitazone int64
rosiglitazone int64
insulin int64
change int64
diabetesMed int64
readmitted int64

```

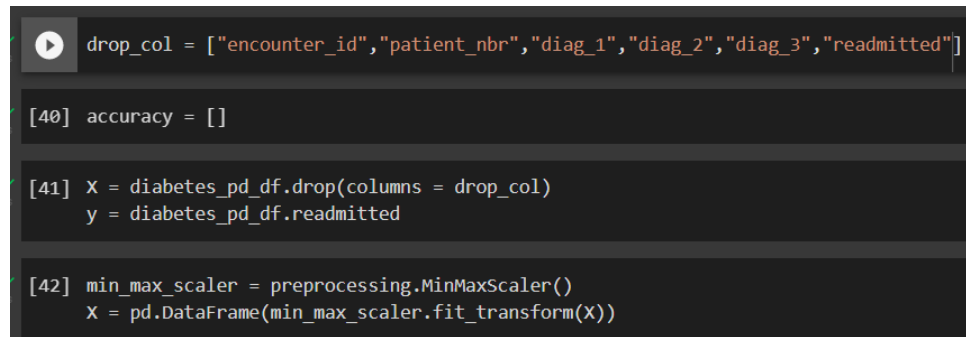
Figure 16: Encoding object data to integer

The diag1 and diag2 columns contained some null value and they were removed since these two columns will be essential for modeling. We created a column with the difference between first and second using the following formula:

$$\text{Change Diag}\% = \frac{(\text{Diag1} - \text{Daig2})}{\text{Daig1}} \times 100$$

We also checked the correlation between the columns but there were no significant results, so we chose not to go with deeper analysis.

Finally, we divided the column into x and y variables and normalized the columns using a min max scaler.



```
drop_col = ["encounter_id", "patient_nbr", "diag_1", "diag_2", "diag_3", "readmitted"]

[40] accuracy = []

[41] x = diabetes_pd_df.drop(columns = drop_col)
    y = diabetes_pd_df.readmitted

[42] min_max_scaler = preprocessing.MinMaxScaler()
    x = pd.DataFrame(min_max_scaler.fit_transform(x))
```

Figure 17: Finalize preprocessing

4. Modeling

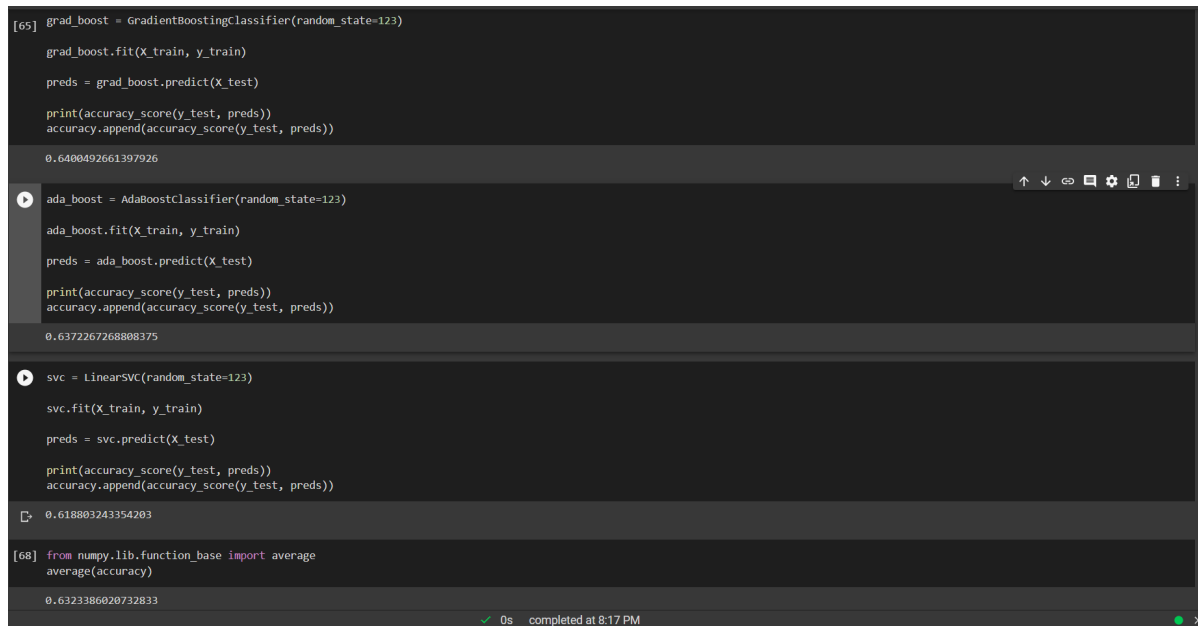
We tested multiple linear, tree and ensemble models and picked the one that gave us the best prediction accuracy. The models that were finalized and their accuracy were as following:

- Random Forest Classifier - 0.6332751719183003
- Gradient Boost Classifier - 0.6400492661397926
- AdaBoost Classifier - 0.6372267268808375
- Linear SCV - 0.618803243354203

As seen from the results, ensemble models gave us the highest accuracy score. Gradient Boost classifier was the best predictor of all models.

5. Split of dataset

The models will be trained using a train test split of 80:20. This was chosen by taking the average accuracy score from all models. We tested models from 50:50 split up to 80:20 split.



```
[65] grad_boost = GradientBoostingClassifier(random_state=123)
      grad_boost.fit(X_train, y_train)
      preds = grad_boost.predict(X_test)
      print(accuracy_score(y_test, preds))
      accuracy.append(accuracy_score(y_test, preds))

0.6400492661397926
```

```
ada_boost = AdaBoostClassifier(random_state=123)
ada_boost.fit(X_train, y_train)
preds = ada_boost.predict(X_test)
print(accuracy_score(y_test, preds))
accuracy.append(accuracy_score(y_test, preds))

0.6372267268808375
```

```
svc = LinearSVC(random_state=123)
svc.fit(X_train, y_train)
preds = svc.predict(X_test)
print(accuracy_score(y_test, preds))
accuracy.append(accuracy_score(y_test, preds))

0.618803243354203
```

```
[68] from numpy.lib.function_base import average
      average(accuracy)

0.6323386020732833
```

0s completed at 8:17 PM

Figure 18: Testing average accuracy for all models with different train test split

Anything higher than 80% training split would lead to issues like overfitting and thus we limited the split to 80%.

Result

Our initial analysis found that diabetes is more prevalent in the ages of 50-90 and affects females more than male. Persons of Caucasian race are more prone to the disease, this could be due to difference in lifestyle and culture in different races. Asian and Hispanic races have lower diagnosis of diabetes of all races under observation. Also, medication is more effective in older ages compared to younger ones.

The readmission rate is higher in older age groups compared to younger age groups. This could be due to more health complications as the body grows older. Younger patients are able to recover quicker and their body is able to heal quickly. The effectiveness of diabetes medicine can

differ on a case by case basis, but if not given the proper medication the readmission rates get much higher.

Finally, our modeling with a train test split of 80-20 gave us the highest prediction accuracy. Gradient Boost Classifier was the best model with an accuracy of 64% for predictions.

References

- [1] Diabetes 130-US hospitals for years 1999-2008 Data Set retrieved from <http://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008#>
- [2] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records,” BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.
- [3] Modeling algorithms referred from: https://scikit-learn.org/stable/user_guide.html