Predictive Modeling of Hospital Readmissions for Diabetes Patients

Vysali Kallepalli | Bhanoday Sai Kastala | Shruti Daundkar

Overview of the Project & Data

This project involves performing in-depth data analysis and developing predictive modeling strategies using a dataset containing readmission information for patients with diabetes. The primary goal is to perform exploratory data analysis (EDA) and machine learning to predict hospital readmissions within the 30-day period.

1. The dataset has 101766 records with 50 attributes with 12 numeric, 30 categorical and 4 text and 3 boolean types.
2. There are unique identifiers in the dataset such as encounter_id and patient_nbr.
3. The dataset has critical patient information like age, patient_nbr (patient identifier), race, admission type, patient's time in hospital, HbA1c result, diagnosis, diabetic medications etc.

More information about ID Mapping: Id Mapping

Exploratory Data Analysis (EDA)

We started our project by using ydataprofiling to generate detailed reports of the dataset. However, after taking a closer look at the generated reports, the data source, and the documentation, we noticed some discrepancies. The initial reports indicated missing values that were not actually present in the dataset. This happened because Pandas converted certain 'None' values to 'nan'. In fact, these 'None' values represent valid categories, not missing or invalid data. As a result, the reports incorrectly flagged these columns as having missing values. To resolve this, we performed a manual analysis and cross-checked with the dataset documentation. This helped us confirm the correct information, ensuring that our understanding of the data is accurate.

1. Data Understanding and Preprocessing

a. What data types are present in the dataset?

- The dataset has numeric, categorical, text, and boolean data types. (object , int64). Out of 50 features, 13 of them are Numerical and 37 of them are Nominal data types.

b. Identify columns with missing data.

- `race, weight, payer_code, medical_specialty, diag_1, diag_2, diag_3` have missing data.

c. What percentage of data is missing in each of these columns?

- There are a total of 101766 records in the dataset.
- `race` has 2273 records missing which is 2.23%
- `weight` has 98569 records missing which is 96.85%
- `payer_code` has 40256 records missing which is 39.5%
- `medical_specialty` has 49949 records missing which is 49%
- `diag_1` has 21 records missing which is 0.02%
- `diag_2` has 358 records missing which is 0.35%
- `diag_3` has 1423 records missing which is 1.39%

d. For columns with missing data, how will you handle them?

- Each column is handled differently.
- `race -` It is impossible to fill it with any value based on the frequency or by calculating measures of central tendency or variability. So we would fill it with "Unknown"

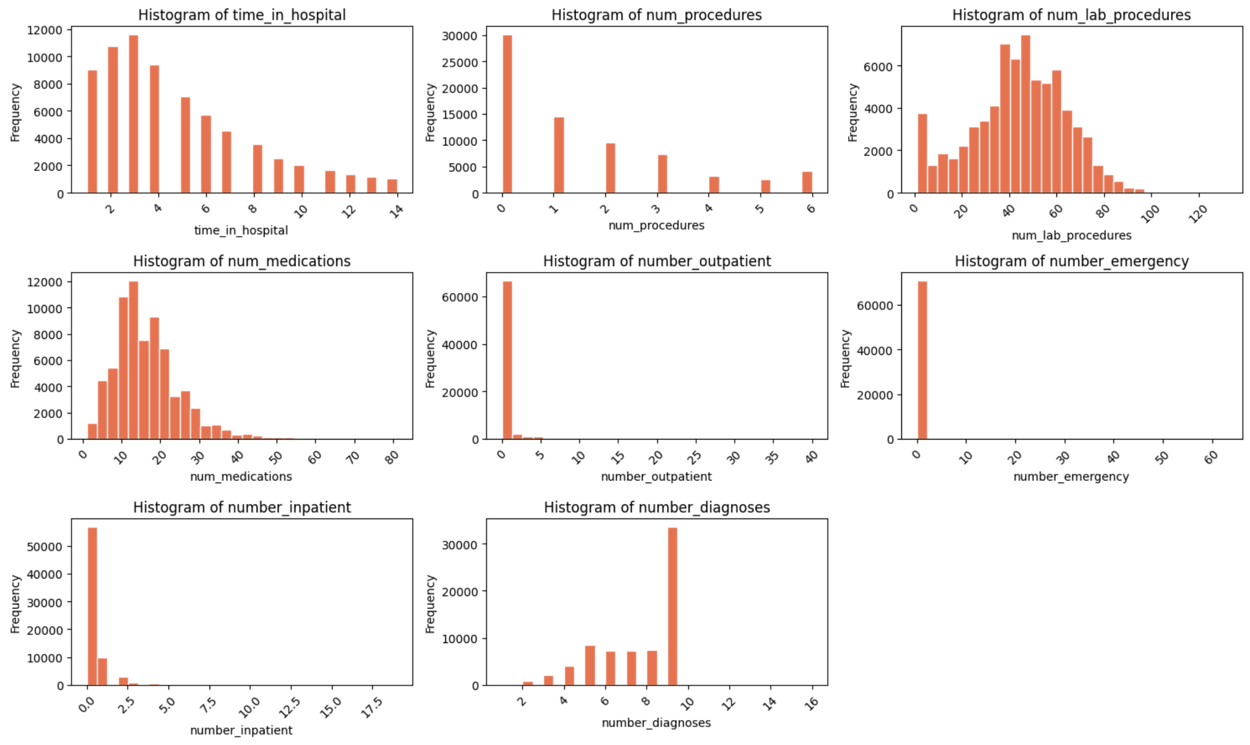
- `weight, payer_code, medical_specialty` - These have a high percentage of missing values, which make them unusable. So we would drop them.
- `diag_1, diag_2, diag_3` - we would just replace it with “Unknown”

❖ Data Cleaning

1. There are attributes in the data with a lot of missing values represented by '?'. we have replaced these values with ‘Unknown’.
2. Removed the identifier ‘`patient_nbr`’since it has duplicate values.
3. Gender has 3 different categories - ‘Male’, ‘Female’, ’Unknown/Invalid’. Eliminated the ’Unknown/Invalid’ entries and now it only has Male and Female.
4. As mentioned above, the weight, medical_specialty and payer_code have a larger percentage of missing values, and they were dropped. (we set a threshold value of 35% and any attribute with more than 35% of the missing data would be dropped)
5. There are a lot of ‘None’ values in max_glu_serum and A1Cresult which means ‘not measured’, but these values are mistaken to be ‘NaN’ values by Pandas, so replacing it with ‘Not tested’
6. From table 2 of this link, we narrowed down the values of diag_1, diag_2 and diag_3 to 9 different categories based on the IC9 code and created 3 different columns for each.
   ★ `Circulatory, Respiratory, Digestive, Diabetes, Injury, Musculoskeletal, Genitourinary, Neoplasms, Other`
7. The below categorical data represents the drug and its doses and have very little variation which makes them zero variance variables, and they do not seem related to the readmissions and are not useful for prediction. So we will not consider them at the moment.
```
metformin                repaglinide    nateglinide       chlorpropamide
glimepiride                     acetohexamide  glipizide          glyburide
tolbutamide                     pioglitazone   rosiglitazone      acarbose
miglitol                 troglitazone   tolazamide        examide
citoglipton                     glyburide-metformin  glipizide-metformin
glimepiride-pioglitazone metformin-rosiglitazone
   metformin-pioglitazone
```
8. The attribute readmitted has 3 values. <30 , >30 and NO. Narrowed these values to 0 and 1. >30 to 0 and <30 and NO to 1. We did this to convert categorical data into numeric values so that it will be used for our machine learning algorithms.

2. Visualization

a. Histograms for key numerical variables. What do these tell you about the distribution of the data



Time_in_hospital : The histogram indicates a moderately short hospital stay, with most patients staying around the mean duration of 4.40 days.

Num_lab_procedures : A wide range and high mean suggest significant variability. The histogram indicates most patients undergo a substantial number of lab procedures.

Num_procedures : Nearly half the patients had no procedures, leading to a right-skewed histogram. When procedures are performed, they are generally few.

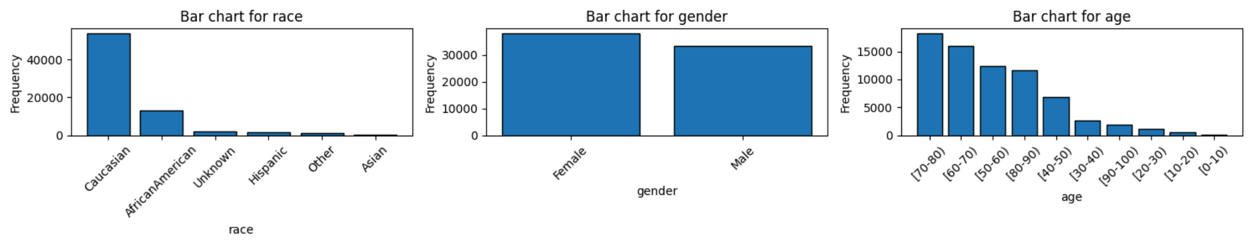
Num_medications : The histogram shows a wide spread with a high mean, indicating many patients are prescribed a significant number of medications.

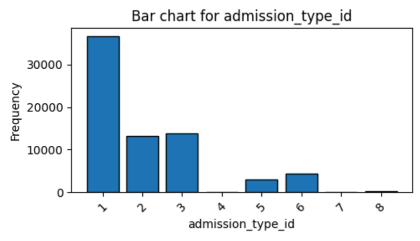
Number_outpatient : The histogram is highly right-skewed which means most patients did not have outpatient visits.

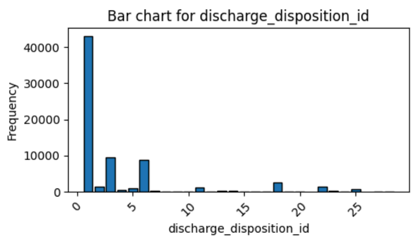
Number_emergency : Similarly to outpatient visits, the histogram is right-skewed with emergency visits being very rare among patients.

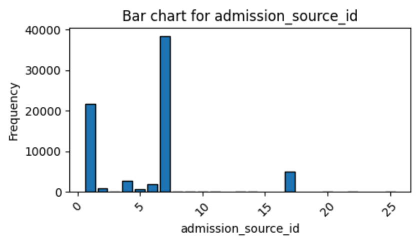
Number_inpatient : The histogram indicates a right-skewed distribution with most patients not having inpatient stays, although less rare than outpatient or emergency visits.

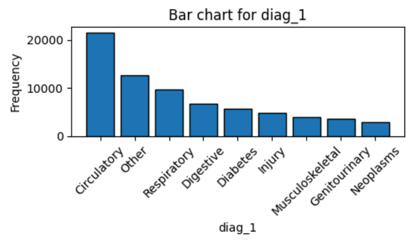
b. Bar charts for key categorical data. What insights can you gather about the population from these charts?

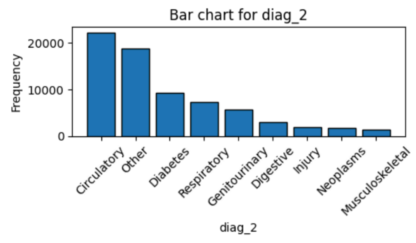


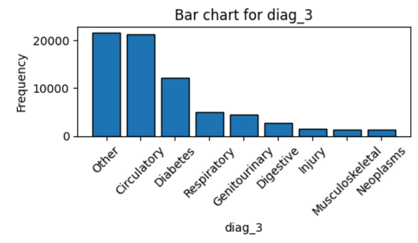


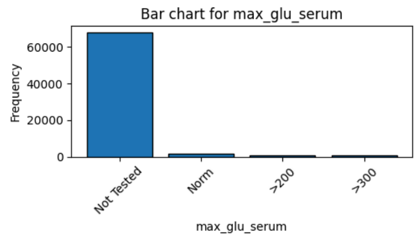


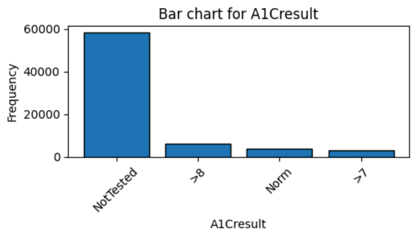


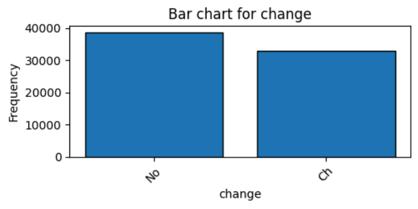


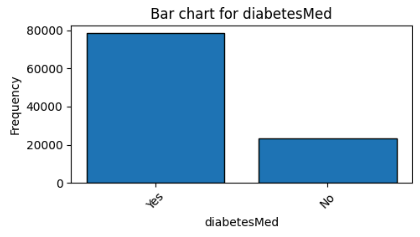


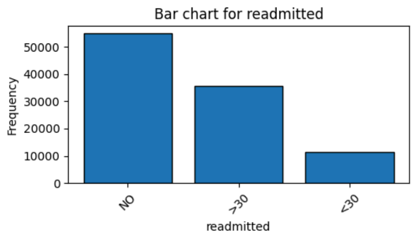












Race: The population has six distinct racial groups including the missing data, but the distribution is uneven, indicating certain races such as Caucasian and AfricanAmerican are more prevalent than others.

Gender: There are two distinct gender categories, with a likely imbalance that suggests a predominance of Female over Male.

Age: The population spans ten age groups, with the highest concentration in the 70-80 age group, indicating that the patient population is largely older adults.

Admission_type_id: The majority of admissions are emergency-related, followed by urgent or elective types.

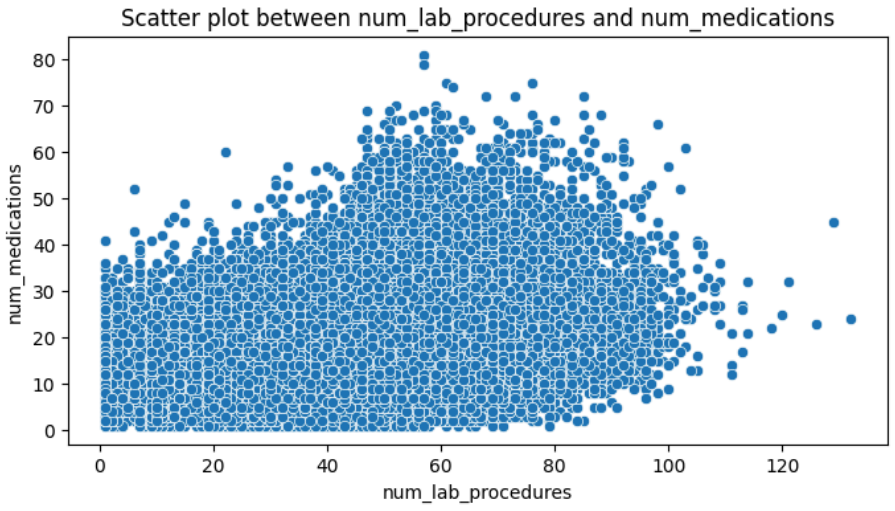
Discharge_disposition_id: Many discharge dispositions have very low frequencies, indicating a long tail distribution. Most patients are discharged to home (value 1 with over 40000 occurrences), but there is a wide range of other discharge outcomes, reflecting post-hospitalization scenarios.

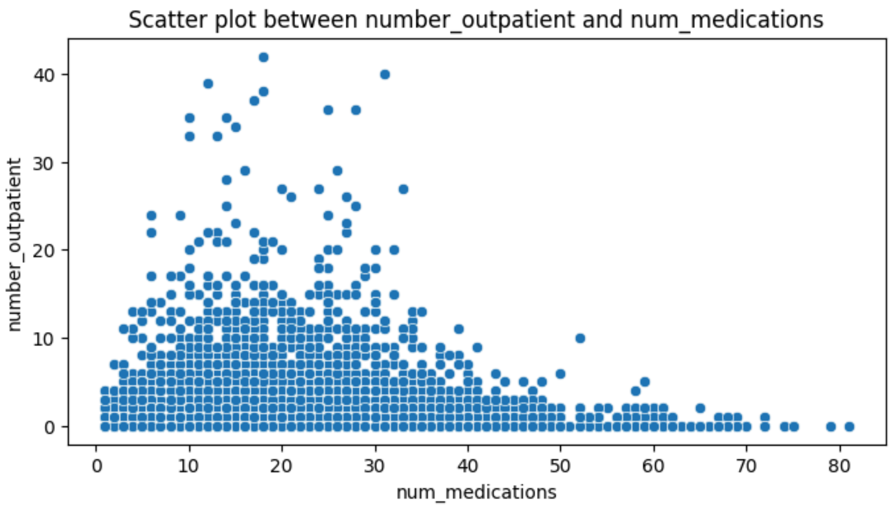
Admission_source_id: The plot shows that a high volume of patients come through emergencies followed by people coming through physician referrals.

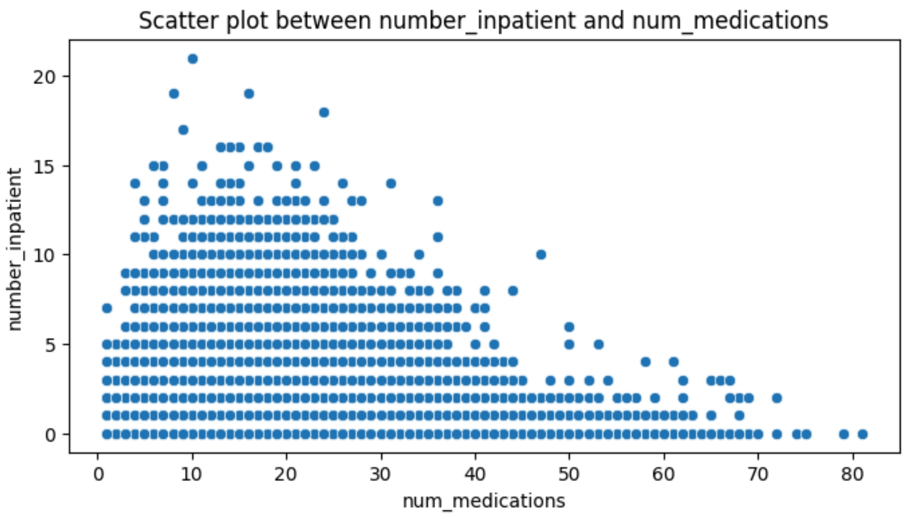
Diag_1 and Diag_2: This represents the diagnosis (first and second) of the patient. From the plot it is evident that most of the patients are diagnosed with 'Circulatory' disease followed by 'Others' (*which includes various diagnoses such as Mental Disorder, Disease of skin etc based on the ic9 code*) 'Respiratory', 'Diabetes' and 'Digestive' problems also being a significant factor.

Diag_3: This represents the third diagnosis of the patient. The plot indicates the majority of the patients are diagnosed with 'Other' category diseases such as Infectious and parasitic diseases, diseases of the blood and blood-forming organs, Congenital anomalies etc. Additionally, Circulatory and Respiratory are also the major factors. (The diagnosis for various ic9 code is referred from Table 2 of this link)

c. **Use scatter plots to examine potential relationships between continuous variables. Are there any noticeable trends?**

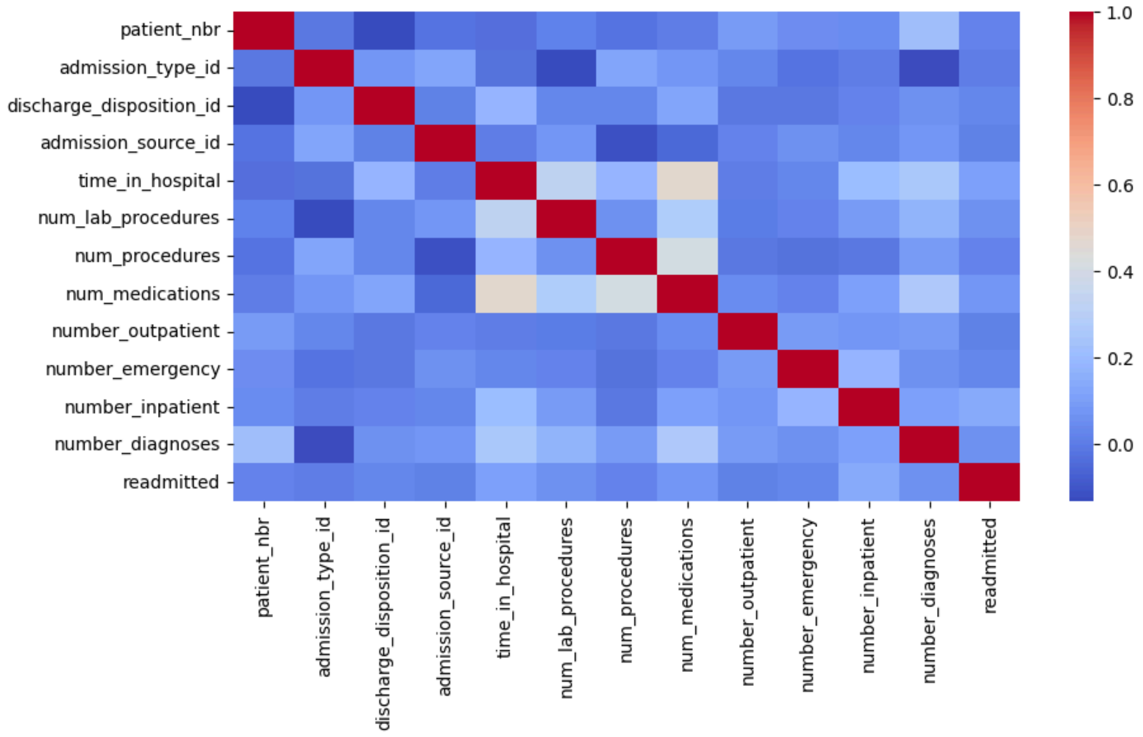






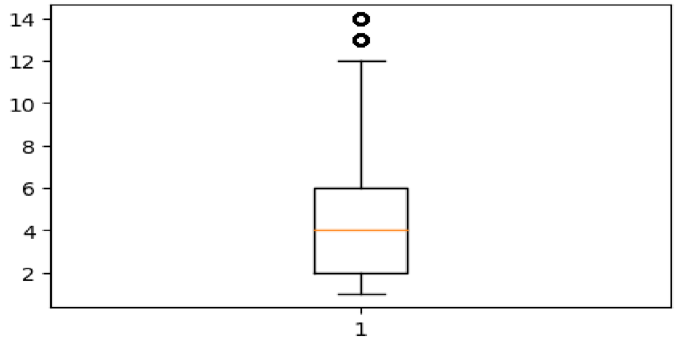
- The scatter plot between the age and time_in_hospital provides a uniform distribution of patients across the different age groups. This says that age may not have a strong correlation with a patient's hospital duration.
- There is a good correlation between the number of lab procedures and the number of medications. This states that as the number of lab procedures increases, the number of medications tends to increase as well. It is possible that more severe or complex conditions require comprehensive treatment.
- An increased number of diagnoses is positively correlated with an extended stay in the hospital. There are some discernible clusters around 9 and 15 diagnoses, indicating patient groups with comparable numbers of diagnoses who have had protracted hospital stays. Patients with many diagnoses may require longer hospital stays, which may be predicted by this trend if they require intensive care or monitoring.
- There is a positive correlation between number_inpatient and num_medications, when the number of inpatient visits rises with the number of prescriptions written. This pattern implies that individuals who are hospitalized more frequently—possibly as a result of serious or persistent health problems—need greater drug management.

d. Create a correlation heatmap of the numerical variables. Which variables show strong positive or negative correlations? How might these relationships affect the prediction of readmissions?



Variables such as `time_in_hospital`, `num_medications`, and `number_diagnoses`, `number_inpatient` exhibit strong positive correlations, indicating that patients with longer hospital stays tend to undergo more procedures, receive more medications, and have more diagnoses.
The moderate positive correlation between `number_inpatient` and `readmitted` suggests that a higher number of inpatient visits is slightly associated with a higher likelihood of readmission. This makes sense as it could reflect more serious or chronic health issues.

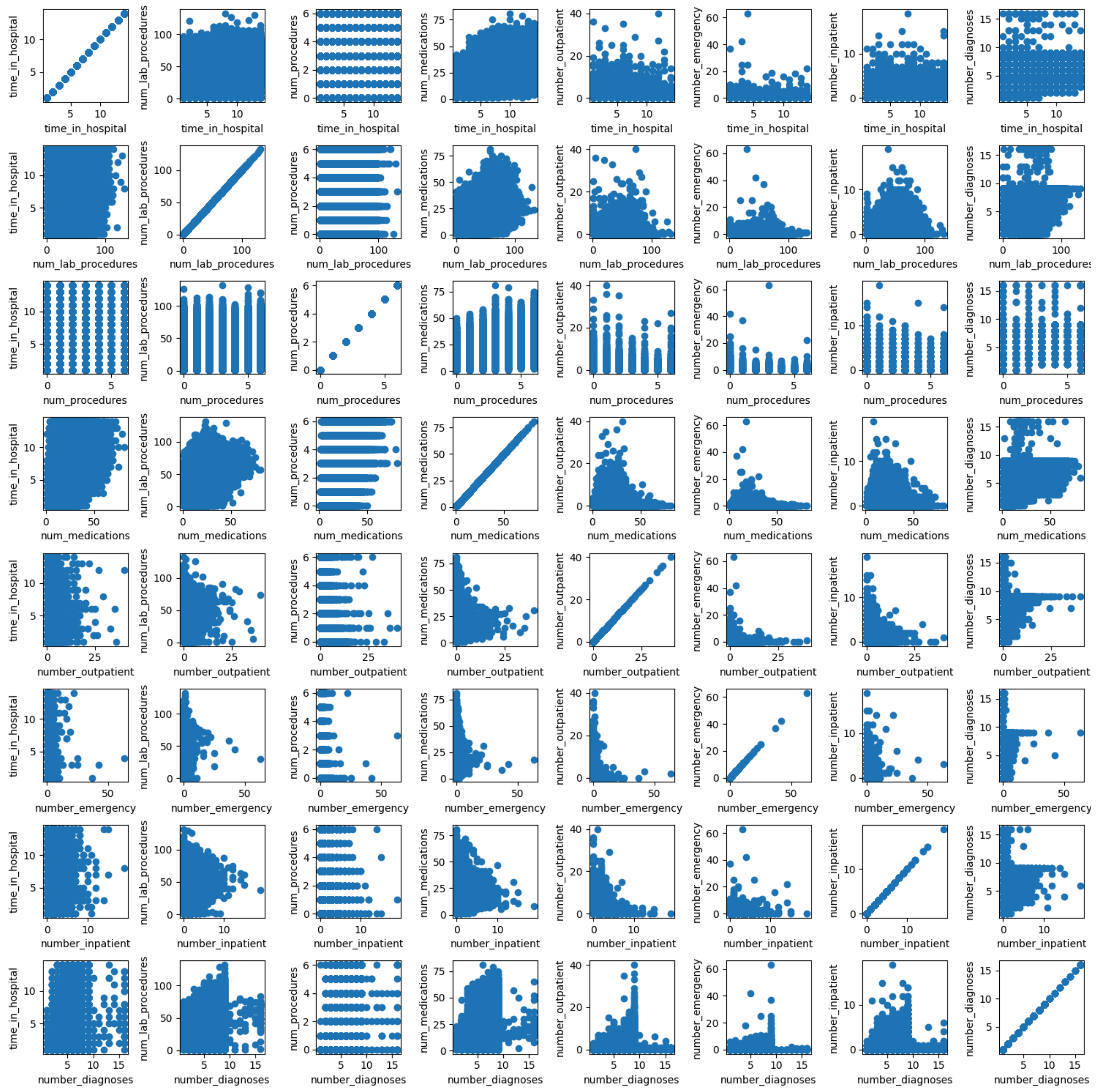
e. Can you identify any outliers in the data? How might these impact your analysis?



There were not many outliers in the data. We validated this by plotting each of the columns. The values 12, 14 whiskers cannot be the outliers as there is a possibility of people staying in hospitals for 12 or 14 days.

f. Are there any multivariable relationships that you want to examine?

We created a matrix of scatter plots for numerical variables in the dataset



Machine Learning Prediction

1. What steps are necessary to make the data suitable for modeling? Discuss any transformations, scaling, or encoding methods you apply.

Before we apply the machine learning models, we need to pre process the data. Below are some of the steps applied:

a. Handling Missing Values
b. Feature Engineering
c. Feature Selection
d. Data Splitting (Test and Training)
e. One Hot Encoding

Handling missing values: It is a crucial process because it can affect the accuracy and introduce bias. In our dataset we have dropped columns such as weight, payer_code, and medical_specialty had missing values.

Feature Engineering: it involves creating new columns or modifying the existing columns based on the algorithm to improve the performance of the model. We have converted the age column to single integer values to simplify the data. As our main focus is on diabetes patients, columns diag_1, diag_2, diag_3 columns have values starting with '250' are categorized as Diabetes and all other values are named as Others. Finally, the target column is transformed into a binary variable with 0 for NO and 1 for <30 and >30.

Feature Selection: It involves choosing the relevant features for the model to reduce the overfitting. We have used the concept of correlation to understand the relation between the target variable and each other. Features with higher correlation such as the medication columns are dropped to avoid overfitting.

Data Splitting: The dataset was split into training and testing sets to evaluate the model's performance. 80% of the data was used to train the model and remaining 20% for the testing and validation.

One-hot encoding: This process is applied to the categorical data to convert them into numerical format that can be used for machine learning algorithms. Columns such as race, gender, age, payer_code, and medical_specialty were one-hot encoded and we dropped the first category to avoid multicollinearity by using drop_first parameter.

2. Choose at least two machine learning algorithms for predicting hospital readmissions. Justify your choice of models.

- We chose Logistic Regression and Random Forest for predicting hospital readmissions. Logistic Regression is a simple and fast model that works well when the relationship between the features and the target variable is linear and the features are roughly normally distributed. It also provides coefficients that can be interpreted as the effect of each feature on the odds of readmission. Random Forest is a more complex model that can capture non-linear relationships and interactions between features. It is also less prone to overfitting than Logistic Regression.

3. Describe how you will split the data into training and testing sets. What proportion of the data will be used for training versus testing?

We will use the train_test_split function from sklearn to split the data into a training set and a testing set. We used 80% of the data for training and 20% for testing. This is a common split that provides a good balance between having enough data to train the model and having enough data to test its performance.

4. What performance metrics will you use to evaluate your models? Explain why these metrics are appropriate for this analysis.

We will use accuracy, precision, recall, and the AUC-ROC score to evaluate the models. Accuracy measures the proportion of correct predictions, but it can be misleading if the classes are imbalanced. Precision and recall provide more information about the performance of the model on the positive class. The AUC-ROC score measures the ability of the model to distinguish between the classes, regardless of the threshold used to classify the predictions.

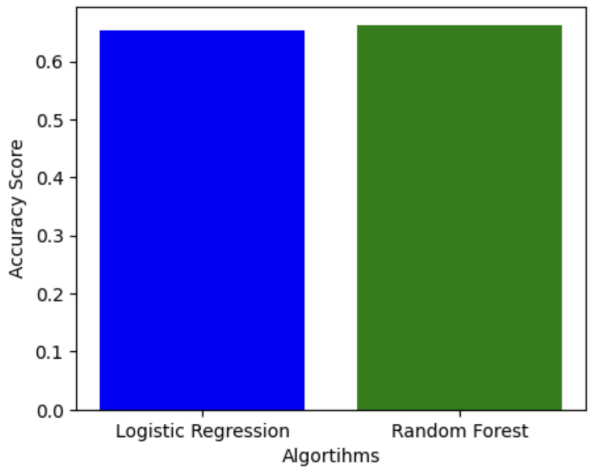
5. How will you interpret the results of these metrics? What will they tell you about the effectiveness of your models?

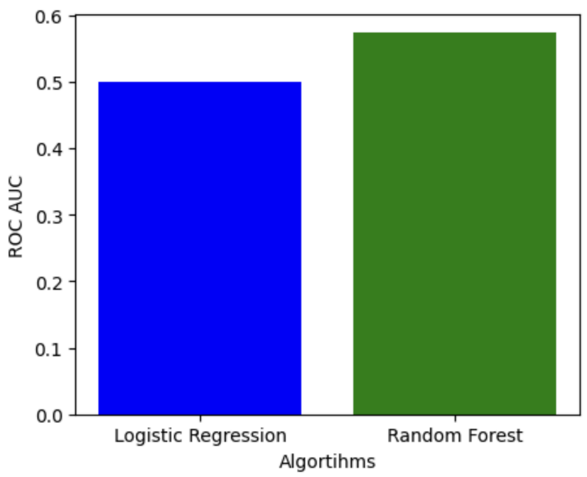
High accuracy means that the model makes correct predictions most of the time, but it doesn't tell us much about its performance on the positive class. High precision means that when the model predicts a readmission, it is usually correct. High recall means that the model is good at identifying readmissions. A high AUC-ROC score means that the model is good at distinguishing between readmissions and non-readmissions.

6. Compare the performance of your models. Based on the results, which model would you recommend for predicting hospital readmissions?

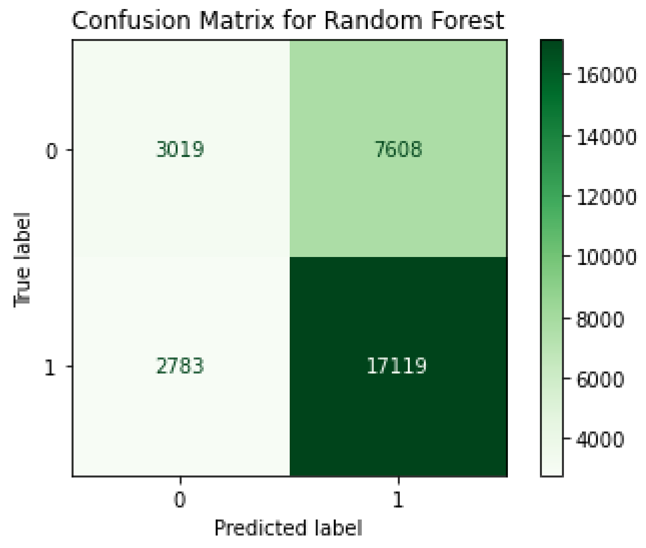
Model Comparison: We will compare the performance of the models on the testing set using the metrics described above. The model with the highest AUC-ROC score would be the best at distinguishing between readmissions and non-readmissions. If the models have similar AUC-ROC scores, we will choose the one with the highest recall, as it is important to identify as many readmissions as possible in this context.

Model Recommendation: Based on the results of the comparison, we best recommend the model that has the highest AUC-ROC score and recall. In this case, it would be Random Forest, this model would be the most effective at predicting hospital readmissions.





Confusion Matrix for Random Forest:



Conclusion

In conclusion, we built two different models to predict or identify diabetic patients who have a higher possibility of being readmitted within 30 days. Based on the preprocessing of data and analysis, we utilized two machine learning algorithms: Logistic Regression and Random Forest. In terms of accuracy and ROC, the Random Forest model emerged as the superior choice, achieving an accuracy of 69%. However, it's worth noting that although initially, we achieved an accuracy of 88%, subsequent adjustments led to a decline in accuracy, which we were unable to recover.

Several key factors were identified as influential for readmission, including the number of diagnoses, the frequency of hospitalizations, the quantity of lab procedures, and the number of medications administered. Additionally, age emerged as a crucial factor, with older patients exhibiting a higher likelihood of readmission in certain cases. Other variables such as insulin usage, admission type, and primary diagnosis also contributed, although to a lesser extent, to the prediction of readmitting patients.

Future Work

- Data preprocessing could be improved further; it is possible that we have dropped columns that could have been useful for the model's performance.
- Instead of dropping columns with missing values, we could have replaced them with the most frequent value.
- Our current chosen algorithm provided satisfactory results, we should explore alternative methods such as Naive Bayes and Decision trees may lead to higher accuracy.

Appendix: Python Code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from ydata_profiling import ProfileReport
get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:

df = pd.read_csv("diabetic_data.csv")


# In[3]:

# Checking for '?' or missing values
missing_val = df.applymap(lambda x: x == "?" or pd.isnull(x))
# Number of invalid entries per column
num_invalid_entries_per_column = missing_val.sum()
print(num_invalid_entries_per_column)


# In[4]:

# Replace 'None' with 'NotTested' for max_glu_serum and A1Cresult
df["max_glu_serum"] = df["max_glu_serum"].replace(np.nan, "Not Tested")
df["A1Cresult"] = df["A1Cresult"].replace(np.nan, "NotTested")


# In[5]:

df["max_glu_serum"].unique()
df["A1Cresult"].unique()


# In[6]:

# Calculating the percentage of missing and invalid data in each column
missing_val = df.applymap(lambda x: x == "?" or pd.isnull(x))
missing_invalid_data = missing_val.sum() / len(df) * 100
print(missing_invalid_data)
```

```
# In[7]:

# Replacing '?' with Other
df = df.replace("?", "Unknown")


# In[8]:

# Dropping columns with more than 35% missing data
df = df.drop(["weight", "payer_code", "medical_specialty"], axis=1)


# In[9]:

# dropping patient_nbr column
df = df.drop(["patient_nbr"], axis=1)


# In[10]:

print(df["gender"].unique())


# In[11]:

# Dropping gender entries with 'Unknown/Invalid'
df = df[df["gender"] != "Unknown/Invalid"]


# In[12]:

# Categorizing diag_1, diag_2, diag_3 into 9 categories based on ICD-9 codes
def categorize_diag(value):
    try:
        if value == "250.xx" or value.startswith("250"):
            return "Diabetes"
        else:
            return "Other"
    except ValueError:
        if value.startswith("E") or value.startswith("V"):
            return "Other"
    return "Other"


df["diag_1"] = df["diag_1"].apply(categorize_diag)
df["diag_2"] = df["diag_2"].apply(categorize_diag)
```

```
df["diag_3"] = df["diag_3"].apply(categorize_diag)


# In[13]:

df["diag_1"].unique()


# In[14]:

numeric = df[
    [
        "time_in_hospital",
        "num_procedures",
        "num_lab_procedures",
        "num_medications",
        "number_outpatient",
        "number_emergency",
        "number_inpatient",
        "number_diagnoses",
    ]
]
categorical = df[
    [
        "race",
        "gender",
        "admission_type_id",
        "discharge_disposition_id",
        "admission_source_id",
        "diag_1",
        "diag_2",
        "diag_3",
        "change",
        "diabetesMed",
        "readmitted",
    ]
]


# In[15]:

# Generate histograms
for var in numeric:
    plt.figure(figsize=(8, 4))
    plt.hist(df[var], bins=30, edgecolor="black")
    plt.title(f"Histogram of {var}")
    plt.xlabel(var)
    plt.ylabel("Frequency")
```

```python
    plt.show()


# In[16]:

for var in categorical:
    plt.figure(figsize=(8, 4))
    df[var].value_counts().plot(kind="bar", edgecolor="black")
    plt.title(f"Bar plot of {var}")
    plt.xticks(rotation=45)
    plt.xlabel(var)
    plt.ylabel("Frequency")
    plt.show()


# In[17]:

# more cleaning
df = df.drop(
    columns=[
        "metformin",
        "repaglinide",
        "nateglinide",
        "chlorpropamide",
        "glimepiride",
        "acetohexamide",
        "glipizide",
        "glyburide",
        "tolbutamide",
        "pioglitazone",
        "rosiglitazone",
        "acarbose",
        "miglitol",
        "troglitazone",
        "tolazamide",
        "examide",
        "citoglipton",
        "glyburide-metformin",
        "glipizide-metformin",
        "glimepiride-pioglitazone",
        "metformin-rosiglitazone",
        "metformin-pioglitazone",
    ],
    axis=1,
)


# In[18]:
```

```
df["readmitted"] = df["readmitted"].replace(">30", 0)
df["readmitted"] = df["readmitted"].replace("<30", 1)
df["readmitted"] = df["readmitted"].replace("NO", 0)


# In[19]:

# remove readmitted from categorical
categorical = categorical.drop(columns=["readmitted"], axis=1)


# In[20]:

categorical

# In[21]:

df.shape

# In[22]:

# MultiVariable Analysis
import matplotlib.pyplot as plt

# df_num_cols = df[['time_in_hospital', 'num_lab_procedures', 'num_procedures',
'num_medications',
#                    'number_outpatient', 'number_emergency', 'number_inpatient',
'number_diagnoses']]

fig, axs = plt.subplots(len(numeric.columns), len(
    numeric.columns), figsize=(15, 15))

for i in range(len(numeric.columns)):
    for j in range(len(numeric.columns)):
        axs[i, j].scatter(numeric[numeric.columns[i]],
                          numeric[numeric.columns[j]])
        axs[i, j].set(xlabel=numeric.columns[i], ylabel=numeric.columns[j])

plt.tight_layout()
plt.show()


# In[23]:

# correlation matrix
# Compute the correlation matrix
corr_matrix = numeric.corr()
```

```
# print the correlation matrix
print(corr_matrix)

# Plot the heatmap
plt.figure(figsize=(10, 5))
sns.heatmap(corr_matrix, cmap="coolwarm")
plt.show()


# In[24]:

age_id = {
    "[0-10)": 5,
    "[10-20)": 15,
    "[20-30)": 25,
    "[30-40)": 35,
    "[40-50)": 45,
    "[50-60)": 55,
    "[60-70)": 65,
    "[70-80)": 75,
    "[80-90)": 85,
    "[90-100)": 95,
}
df["age"] = df.age.replace(age_id)


# In[25]:

categorical["diag_1"].unique()


# In[26]:

categorical = pd.get_dummies(categorical)


# In[27]:

categorical = categorical.astype("int64")


# In[28]:

categorical


# In[29]:
```

```
dff = pd.concat(
    [df[["encounter_id", "age", "readmitted"]], categorical, numeric], axis=1
)


# In[30]:

dff.columns


# In[31]:

dff.head()


# In[32]:

dff = dff.drop(
    columns=["diag_1_Other", "diag_2_Other", "diag_3_Other"], axis=1)


# In[33]:

# importing libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
roc_auc_score


# In[34]:

# replace 'readmitted' with your actual target column
X = dff.drop("readmitted", axis=1)
y = dff["readmitted"]  # replace 'readmitted' with your actual target column

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)


# In[35]:

from sklearn.metrics import confusion_matrix, accuracy_score, make_scorer
from sklearn.model_selection import cross_validate
```

```


def tn(y_true, y_pred):
    return confusion_matrix(y_true, y_pred)[0, 0]


def fp(y_true, y_pred):
    return confusion_matrix(y_true, y_pred)[0, 1]


def fn(y_true, y_pred):
    return confusion_matrix(y_true, y_pred)[1, 0]


def tp(y_true, y_pred):
    return confusion_matrix(y_true, y_pred)[1, 1]


# cross validation purpose
scoring = {"accuracy": make_scorer(accuracy_score), "prec": "precision"}
scoring = {
    "tp": make_scorer(tp),
    "tn": make_scorer(tn),
    "fp": make_scorer(fp),
    "fn": make_scorer(fn),
}


def display_result(result):
    print("TP: ", result["test_tp"])
    print("TN: ", result["test_tn"])
    print("FN: ", result["test_fn"])
    print("FP: ", result["test_fp"])


# In[36]:

acc = []
roc = []

clf = LogisticRegression(max_iter=500)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# find accuracy
ac = accuracy_score(y_test, y_pred)
acc.append(ac)
```

```
# find the ROC_AOC curve
rc = roc_auc_score(y_test, y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac, rc))

# cross val score
result = cross_validate(clf, X_train, y_train, scoring=scoring, cv=10)
display_result(result)
pd.DataFrame(data={"Actual": y_test, "Predicted": y_pred}).head()


# In[37]:

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# find accuracy
ac = accuracy_score(y_test, y_pred)
acc.append(ac)

# find the ROC_AOC curve
rc = roc_auc_score(y_test, y_pred)
roc.append(rc)
print("\nAccuracy {0} ROC {1}".format(ac, rc))

# cross val score
result = cross_validate(clf, X_train, y_train, scoring=scoring, cv=10)
display_result(result)
pd.DataFrame(data={"Actual": y_test, "Predicted": y_pred}).head()
```