

TABLE OF CONTENTS

1. Introduction.....	2
2. Aims and Objectives.....	2
Aim.....	2
Objectives.....	2
3. Part 1: Building up a Basic Predictive Model	2
3.1 Data Cleaning and Transformation	2
3.2 Data Visualisation	6
3.3 Model Building.....	10
4. Part 2: Improved Model.....	11
5. Conclusion	14
Reference List.....	15

1. Introduction

This report presents the results of a data science project to build a classification model to predict whether a patient will be readmitted to a hospital within 30 days using diabetes data from 130 US hospitals. The datasets known as the Diabetes Database and spans ten years (1999-2008) and includes 47 characteristics and 101,766 cases. The project involves data cleaning and research and visualization and development of predictive models, including advanced models and cluster-based classification models.

2. Aims and Objectives

Aim

The main aim of this project is to determine the early readmission of patients within 30 days of discharge by developing a robust predictive model using the proposed diabetes dataset.

Objectives

The specific objectives are:

- To prepare data and clean data with the question to be answered in mind.
- To build a basic forecasting model and evaluate its performance.
- To develop a classification of advanced models. model with higher performance and as many data points as possible.
- To implement a cluster-based classification model using the K Means algorithm and compare its performance with global and advanced models.

3. Part 1: Building up a Basic Predictive Model

```
# Loading the dataset
diabetic_data = pd.read_csv('/content/drive/MyDrive/data/diabetic_data (1).csv')
```

Figure 1: Loading the Dataset

(Source: Self-Created in Google Colab)

The first step of the project involved loading the diabetes dataset (diabetic_data.csv) into the pandas dataframe as shown in Figure 1.

3.1 Data Cleaning and Transformation

The data cleaning and transformation process involved several steps to handle missing values and remove irrelevant columns and prepare the data for modeling.

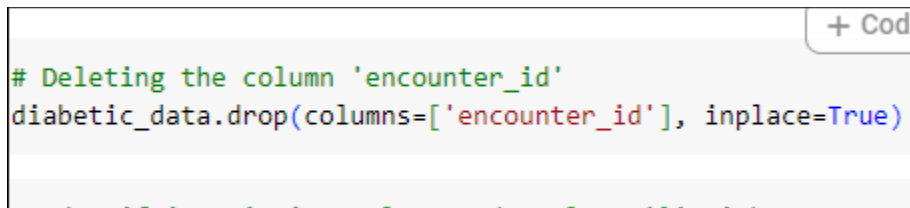
```
# shape of the data
print("Shape of the data:", diabetic_data.shape)

Shape of the data: (101766, 50)
```

Figure 2: Shape of the Data

(Source: Self-Created in Google Colab)

Figure 2 shows the original shape of the data and 101 and 766 rows and 50 columns.

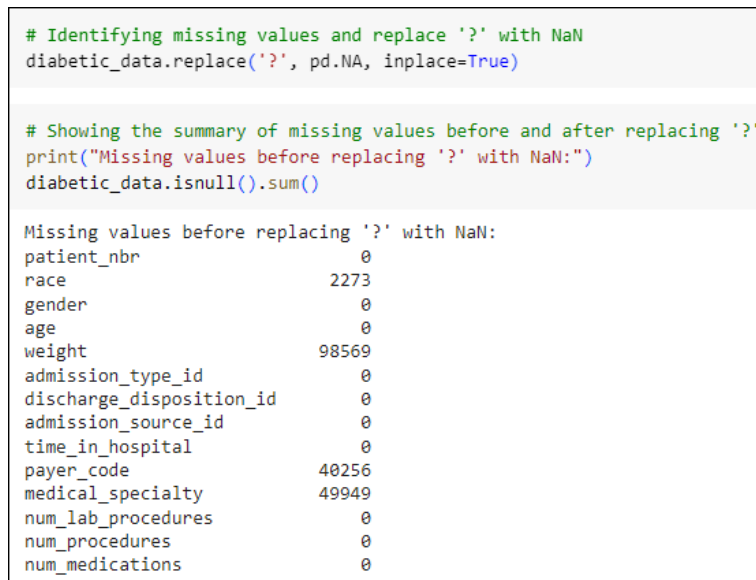


```
# Deleting the column 'encounter_id'
diabetic_data.drop(columns=['encounter_id'], inplace=True)
```

Figure 3: Deleting the 'encounter_id' column

(Source: Self-Created in Google Colab)

The Encounter_id column was removed if it was not relevant to the analysis as shown in the Figure 3.



```
# Identifying missing values and replace '?' with NaN
diabetic_data.replace('?', pd.NA, inplace=True)

# Showing the summary of missing values before and after replacing '?'
print("Missing values before replacing '?' with NaN:")
diabetic_data.isnull().sum()

Missing values before replacing '?' with NaN:
patient_nbr      0
race             2273
gender           0
age              0
weight          98569
admission_type_id      0
discharge_disposition_id      0
admission_source_id      0
time_in_hospital      0
payer_code        40256
medical_specialty    49949
num_lab_procedures      0
num_procedures         0
num_medications        0
```

Figure 4: Identifying missing values and replacing '?' with NaN

(Source: Self-Created in Google Colab)

Missing values from columns marked with '?' character was detected and replaced with NaN as shown in Figure 4.

```
# Checking the datatype of each column
print("Datatype of each column:")
diabetic_data.dtypes

Datatype of each column:
patient_nbr          int64
race                 object
gender               object
age                 object
weight              object
admission_type_id    int64
discharge_disposition_id int64
admission_source_id  int64
time_in_hospital     int64
payer_code           object
medical_specialty    object
num_lab_procedures   int64
num_procedures        int64
num_medications       int64
number_outpatient     int64
number_emergency      int64
number_inpatient      int64
diag_1               object
diag_2               object
diag_3               object
number_diagnoses      int64
max_glu_serum        object
```

Figure 5: Data Type of Each Column

(Source: Self-Created in Google Colab)

Figure 5 illustrates the data types of each column that were checked. Additionally the target variable 'reameid' has been converted to a binary attribute and replacing the values '30' and 'NO' with zero.

```
# Calculating the percentage of missing values for each column
missing_percentage = (diabetic_data.isnull().sum() / len(diabetic_data)) * 100
missing_percentage

patient_nbr          0.000000
race                 2.233555
gender               0.000000
age                 0.000000
weight              96.858479
admission_type_id    0.000000
discharge_disposition_id 0.000000
admission_source_id  0.000000
time_in_hospital     0.000000
payer_code           39.557416
medical_specialty    49.082208
num_lab_procedures   0.000000
num_procedures        0.000000
num_medications       0.000000
number_outpatient     0.000000
number_emergency      0.000000
number_inpatient      0.000000
diag_1               0.020636
diag_2               0.351787
diag_3               1.398306
number_diagnoses      0.000000
max_glu_serum        0.000000
```

Figure 6: Calculating Percentage of missing values for each row

(Source: Self-Created in Google Colab)

For each column the percentage of missing values was calculated as in Figure 6. Columns with more than 90% missing values were discarded.

```
# Dropping the columns with more than 90% missing values
columns_to_drop = missing_percentage[missing_percentage > 90].index
diabetic_data.drop(columns=columns_to_drop, inplace=True)

# Dropping the columns with no variations
diabetic_data.drop(columns=['examide', 'citoglipton'], inplace=True)

# Dropping rows with null values
diabetic_data.dropna(inplace=True)

# Defining the list of near-zero-variance columns that need to be deleted
columns_to_delete = ['repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride',
                    'acetoexamide', 'tolbutamide', 'acarbose', 'miglitol',
                    'troglitazone', 'tolazamide', 'glyburide-metformin', 'glipizide-metformin',
                    'glimepiride-pioglitazone', 'metformin-rosiglitazone',
                    'metformin-pioglitazone']

# Dropping the specified columns from the DataFrame
diabetic_data.drop(columns=columns_to_delete, inplace=True)
```

Figure 7: Dropping Columns with missing values and unwanted columns

(Source: Self-Created in Google Colab)

Columns with no variation because "examine" and "cytoglypton" and other columns with almost zero variance removed as in Figure 7.

```
# summary statistics of numerical columns
print("Summary statistics of numerical columns:")
diabetic_data.describe()
```

	patient_nbr	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	num_procedures	num_medications	num...
count	2.675500e+04	26755.000000	26755.000000	26755.000000	26755.000000	26755.000000	26755.000000	26755.000000	numb...
mean	5.732865e+07	2.019137	2.959821	4.972267	4.317922	40.756681	1.475575	16.246720	
std	3.710987e+07	0.946664	4.327149	3.522615	2.949791	19.965546	1.749729	8.609362	
min	7.290000e+02	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	
25%	2.425177e+07	1.000000	1.000000	1.000000	2.000000	29.000000	0.000000	10.000000	
50%	4.401274e+07	2.000000	1.000000	7.000000	4.000000	42.000000	1.000000	15.000000	
75%	9.230198e+07	3.000000	3.000000	7.000000	6.000000	54.000000	2.000000	20.000000	
max	1.893659e+08	6.000000	28.000000	22.000000	14.000000	132.000000	6.000000	81.000000	

Figure 8: Summary Statistics

(Source: Self-Created in Google Colab)

Figure 8 shows rows with zero values have been removed from the dataset. In addition, the summary statistics for the numeric columns were reviewed as shown below to identify and remove outliers from the data.

```
# Printing the shape of resulting dataframe
diabetic_data.shape

(26755, 31)
```

Figure 9: Shape of the resulting data frame

(Source: Self-Created in Google Colab)

If required and feature normalization was performed. The final shape of the resulting dataframe is shown in the Figure 9.

Justification:

For this I have used method such as replacing the missing value, droppping some irrelevant columns and converting the categorical variables to numeric, also changing null values. Dropping columnsn was done in order to reduce noise and focus on relevant information. Converting values was done for future puropose where numerical inputs will be required.

All this techniques we have used were essential for preparing the data, helping us ensure to data completeness reducing demension and transforming data in format used by machine learning algorithm.

3.2 Data Visualisation

After the initial data cleaning and the project involved exploring relationships between various features of the dataset through visualizations.

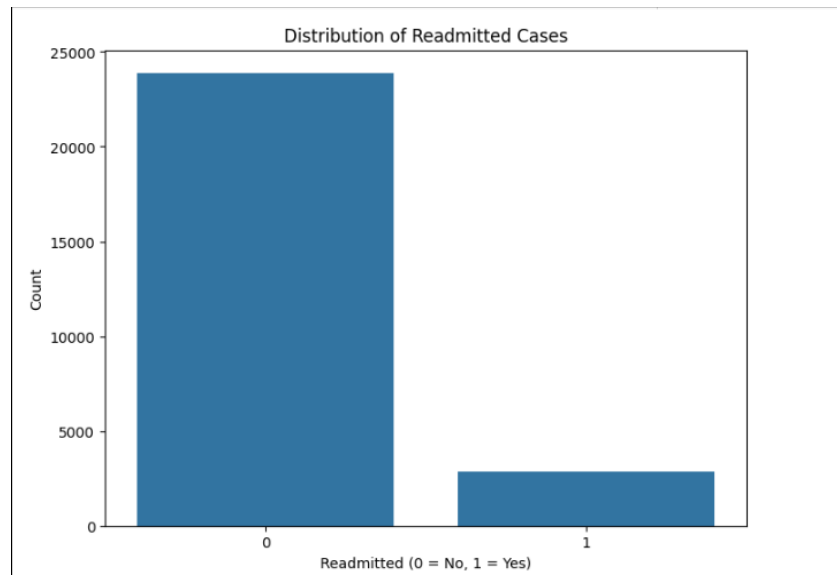


Figure 10: Plot to Analyse the Distribution of Readmitted Cases

(Source: Self-Created in Google Colab)

The distribution of unique classes of the target variable and 'readmitted and' was plotted and as shown in Figure 10.

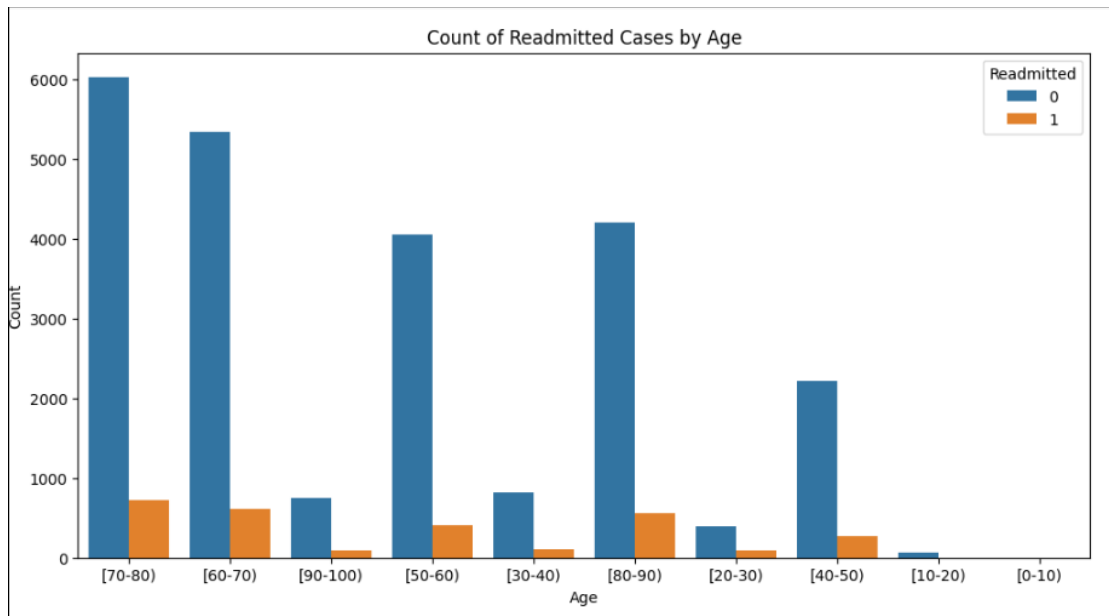


Figure 11: Plot to Analyse the Count of Readmitted Cases by Age

(Source: Self-Created in Google Colab)

The count of readmitted cases was plotted against age to analyze the relationship between these two variables and as depicted in Figure 11.

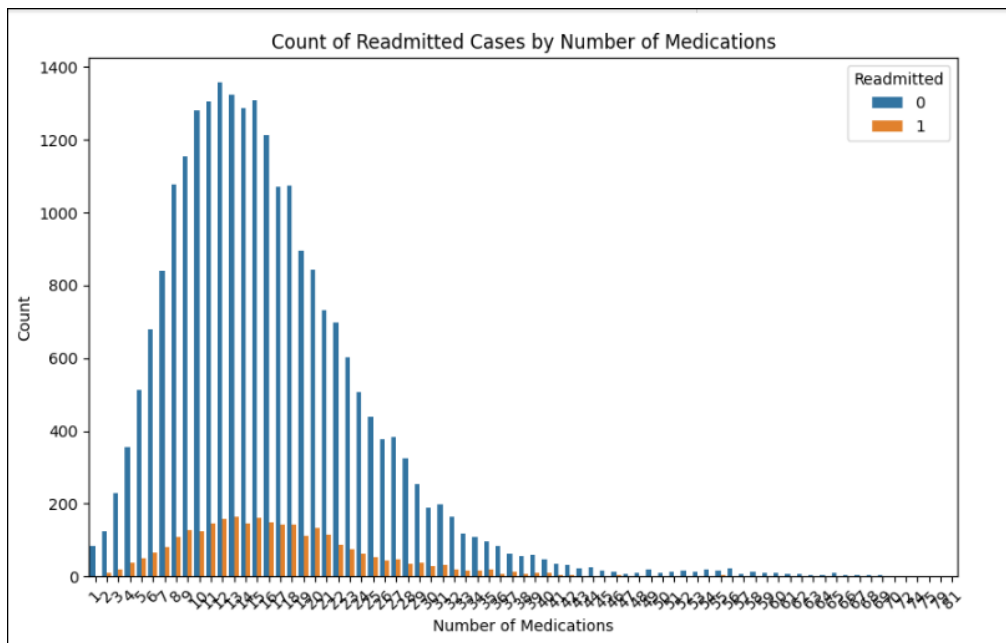


Figure 12: Plot to Count the Readmitted Cases by Number of Medications

(Source: Self-Created in Google Colab)

The count of readmitted cases was plotted against the number of medications to explore the relationship between these variables and as shown in Figure 12.



Figure 13: Scatter Matrix Plot

(Source: Self-Created in Google Colab)

A scatter matrix plot to identify highly correlated feature pairs as shown in Figure 13.

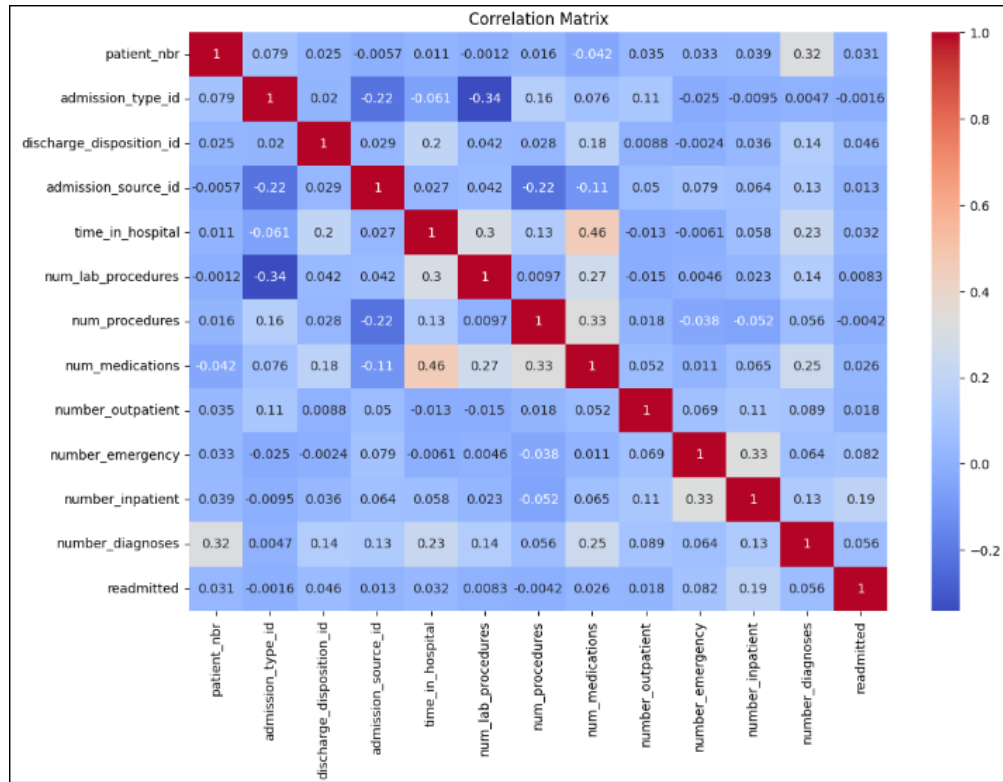


Figure 14: Correlation Matrix

(Source: Self-Created in Google Colab)

A correlation matrix to identify highly correlated feature pairs as shown in Figure 14.

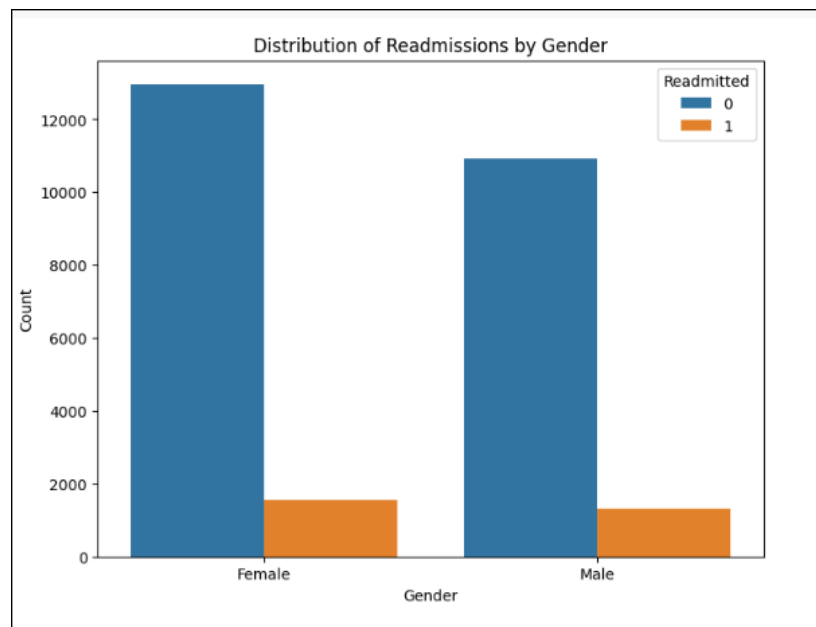


Figure 15: Plot to analyse the Distribution of Readiness by Gender

(Source: Self-Created in Google Colab)

Distribution of readmission by gender and were created to demonstrate an understanding of the problem and the data and as shown in the figure 15.

Justification:

We have used libraries like: Matplotlib and Seaborn. This libraries were used because they provide a wide range of of visulisation tools.

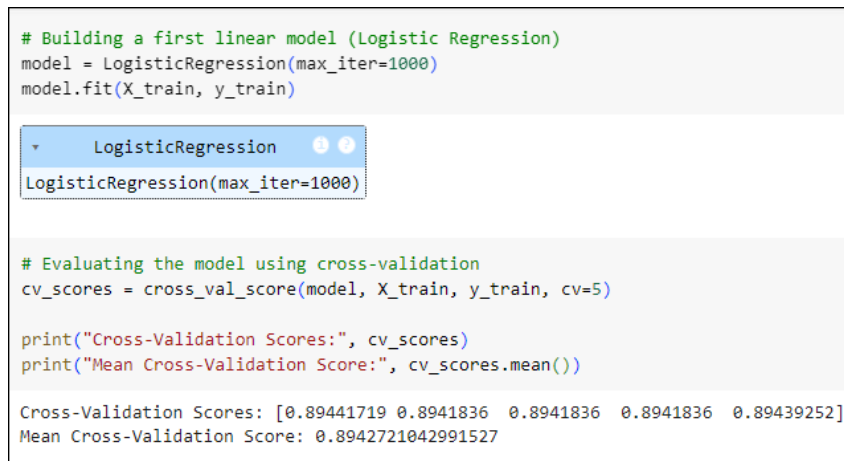
Matplotlib: **plt.hist()** this methods were used to create histograms for analysing the distribution. and **plt.scatter()**- was used to scrate scatter plots for visulising relationships between different variables.

Seaborn- **sns.countplot()**- was used to plot the ocunt of readmitted cases based on the genders, and **sns.pairplot()**- was used to create a scatter matrix plot for identifying highly correlated features.

This libraries and methods used were important in order to explore data distribution and detecting patterns. So, I think using this libraries and methods/techniques was very important to explore data distribution and identifying outliers and understanding the data patterns and.

3.3 Model Building

The basic predictive model was built using the following steps:



```
# Building a first linear model (Logistic Regression)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Evaluating the model using cross-validation
cv_scores = cross_val_score(model, X_train, y_train, cv=5)

print("Cross-Validation Scores:", cv_scores)
print("Mean Cross-Validation Score:", cv_scores.mean())
```

Cross-Validation Scores: [0.89441719 0.8941836 0.8941836 0.8941836 0.89439252]
Mean Cross-Validation Score: 0.8942721042991527

Figure 16: Building the Linear Model and Evaluating Cross Validation

(Source: Self-Created in Google Colab)

Relevant predictors that could impact the prediction of readmission were selected. A linear model was built with these predictors and evaluated using a cross-validation procedure as shown in Figure 16.

Confusion Matrix:					
[[4757 0]					
[594 0]]					
Classification Report:					
	precision	recall	f1-score	support	
0	0.89	1.00	0.94	4757	
1	0.00	0.00	0.00	594	
accuracy			0.89	5351	
macro avg	0.44	0.50	0.47	5351	
weighted avg	0.79	0.89	0.84	5351	

Figure 17: Evaluation of the Model

(Source: Self-Created in Google Colab)

Different performance metrics were used to evaluate the model's performance and considering the imbalanced nature of the data (approximately 10% positive examples) and as shown in Figure 17. The data was balanced using undersampling or oversampling techniques and the model was trained again to achieve better prediction accuracies.

4. Part 2: Improved Model

The second part of this project involved developing an improved classification model with higher performance while utilizing a maximum number of data points.

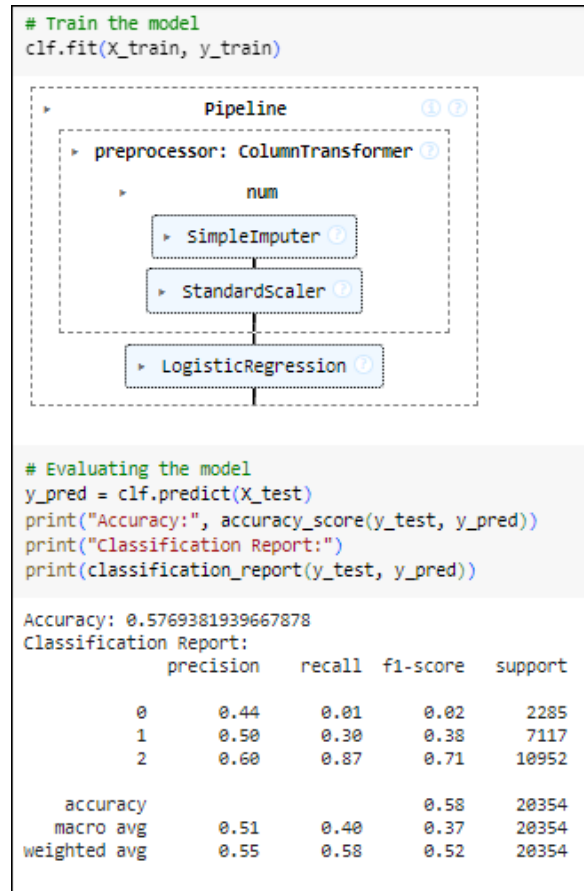


Figure 18: Training and Evaluation of the Improved Model

(Source: Self-Created in Google Colab)

The entire dataset was considered again and missing values were treated differently and such as through imputation and rather than dropping them. Figure 18 below shows the training and evaluation of the improved model.

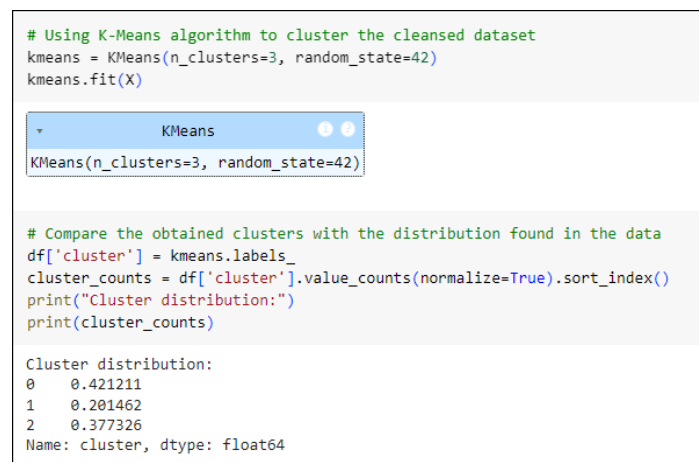


Figure 19: K-Means Algorithm

(Source: Self-Created in Google Colab)

K Means algorithm suite was used to collect the cleaned data and the resulting clusters were compared to the distribution found in the data. The clustering process and results are shown in Figure 19. Clusters were initialized and sets were visualized appropriately.

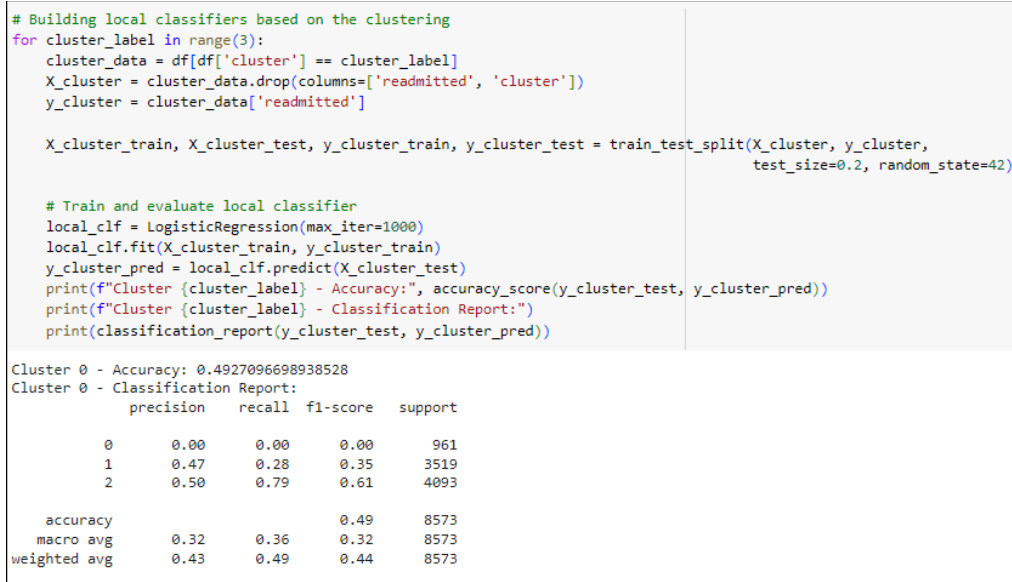


Figure 20: Building and Evaluating Local Classifier Models

(Source: Self-Created in Google Colab)

Local classifiers were built based on the clusters and the performance was compared with the model obtained in the first part of the improved model. Figure 20 shows the process of building and evaluating local classification models. The data has been balanced and the improved model has been trained and tested on the balanced data.

Justification:

For this, we have used models like linear regression, K-means clustering and local classifier. All these were essential as linear regression was used to produce a basic predictive model. K-means clustering was used to group similar data points. And local classifier models were used to utilize more accurate prediction. It helped us in identifying data clustering and local classifier models offering a diverse and comprehensive modelling approach.

For evaluation, cross validation and performance metric was used for producing a robust model performance evaluation and to provide insights of model effectiveness which helps leading into prediction quality. We have used cross-validation technique because it helps us to ensure model generalised. Whereas, performance metric allowed us to provide a qualitative assessment and model accuracy.

5. Conclusion

Hence, it can be concluded that this assignment developed a classification model to predict patient readmission within 30 days using a diabetes database from 130 US hospitals. The process included data cleaning and research and visualization and development of a basic prediction model and an advanced model and a cluster based classification model. The report presented a step by step approach and including data cleaning techniques and visualizations. . understand the data and evaluate the performance of the models. The improved model and the cluster based classification model were designed to achieve better performance by using more data points and taking into account imbalances in the dataset. Findings and insights from this project can be valuable in health analytics and helping to identify factors that affect a patient. readmissions and potentially guide interventions to improve patient outcomes and reduce healthcare costs.

Reference List

Journals

Betty Jane, J. and Ganesh, E.N., 2020. Big data and internet of things for smart data analytics using machine learning techniques. In *Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCB-2019)* (pp. 213-223). Springer International Publishing.

Bowles, M., 2019. *Machine Learning with Spark and Python: Essential Techniques for Predictive Analytics*. John Wiley & Sons.

Doleck, T., Lemay, D.J., Basnet, R.B. and Bazalais, P., 2020. Predictive analytics in education: a comparison of deep learning frameworks. *Education and Information Technologies*, 25, pp.1951-1963.

Raschka, S., Patterson, J. and Nolet, C., 2020. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11(4), p.193.

Sahoo, K., Samal, A.K., Pramanik, J. and Pani, S.K., 2019. Exploratory data analysis using Python. *International Journal of Innovative Technology and Exploring Engineering*, 8(12), pp.4727-4735.

Souza, J., Leung, C.K. and Cuzzocrea, A., 2020. An innovative big data predictive analytics framework over hybrid big data sources with an application for disease analytics. In *Advanced Information Networking and Applications: Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA-2020)* (pp. 669-680). Springer International Publishing.