AI BASED DIABETES PREDICTION SYSTEM

Phase 3: Development Part 1

# Todo:

* Load the dataset
* Preprocess it
* Perform different analysis as needed
* Document all above steps

# Step 1: Importing Libraries

*We will start by importing all the necessary libraries needed.*

*# Importing necessary libraries*

import numpy as np import pandas as pd

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler import matplotlib.pyplot as plt

import seaborn as sns

# Step 2: Loading the Dataset

*Next we will load the diabetes dataset from the provided Kaggle link into our Jupyter Notebook. This dataset contains relevant medical features and information about diabetes.*

*# Load the diabetes dataset from the URL*

dataset\_url = "/content/dataset/diabetes.csv" data = pd.read\_csv(dataset\_url)

# Step 3: Data Exploration

*We'll conduct initial data exploration to understand the structure and characteristics of the dataset. This includes examining data types, summary statistics, and detecting missing values.*

*# Display the first few rows of the dataset*

data.head()

*# Get an overview of the dataset*

data.info()

1. BMI 768 non-null
2. DiabetesPedigreeFunction 768 non-null

float64 float64

*# Check for missing values*

data.isnull().sum()

*# Summary statistics*

data.describe()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 768 entries, 0 to 767 Data columns (total 9 columns):

# Column Non-Null Count Dtype

1. Pregnancies 768 non-null int64
2. Glucose 768 non-null int64
3. BloodPressure 768 non-null int64
4. SkinThickness 768 non-null int64
5. Insulin 768 non-null int64
6. Age 768 non-null int64
7. Outcome 768 non-null int64 dtypes: float64(2), int64(7)

memory usage: 54.1 KB

Pregnancies Glucose BloodPressure SkinThickness Insulin \

count 768.000000 768.000000 768.000000 768.000000

768.000000

mean 3.845052 120.894531 69.105469 20.536458

79.799479

std 3.369578 31.972618 19.355807 15.952218

115.244002

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 0.000000 |  |  |  |  |
| 25% | 1.000000 | 99.000000 | 62.000000 | 0.000000 |
| 0.000000 |  |  |  |  |

50% 3.000000 117.000000 72.000000 23.000000

30.500000

75% 6.000000 140.250000 80.000000 32.000000

127.250000

max 17.000000 199.000000 122.000000 99.000000

846.000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BMI | DiabetesPedigreeFunction | Age | Outcome |
| count | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| mean | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| std | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| min | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| 25% | 27.300000 | 0.243750 | 24.000000 | 0.000000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 50% | 32.000000 | 0.372500 | 29.000000 | 0.000000 |
| 75% | 36.600000 | 0.626250 | 41.000000 | 1.000000 |
|  | max | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

<google.colab.\_quickchart\_helpers.SectionTitle at 0x7ab2d83eace0> import numpy as np

from google.colab import autoviz

def histogram(df, colname, num\_bins=20, figscale=1):

from matplotlib import pyplot as plt

df[colname].plot(kind='hist', bins=num\_bins, title=colname, figsize=(8\*figscale, 4\*figscale))

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = histogram(\_df\_0, \*['Pregnancies'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def histogram(df, colname, num\_bins=20, figscale=1):

from matplotlib import pyplot as plt

df[colname].plot(kind='hist', bins=num\_bins, title=colname, figsize=(8\*figscale, 4\*figscale))

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = histogram(\_df\_1, \*['Glucose'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def histogram(df, colname, num\_bins=20, figscale=1):

from matplotlib import pyplot as plt

df[colname].plot(kind='hist', bins=num\_bins, title=colname, figsize=(8\*figscale, 4\*figscale))

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = histogram(\_df\_2, \*['BloodPressure'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def histogram(df, colname, num\_bins=20, figscale=1):

from matplotlib import pyplot as plt

df[colname].plot(kind='hist', bins=num\_bins, title=colname, figsize=(8\*figscale, 4\*figscale))

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = histogram(\_df\_3, \*['SkinThickness'], \*\*{}) chart

<google.colab.\_quickchart\_helpers.SectionTitle at 0x7ab2d83eac20> import numpy as np

from google.colab import autoviz

def scatter\_plot(df, x\_colname, y\_colname, figscale=1, alpha=.8):

from matplotlib import pyplot as plt

plt.figure(figsize=(6 \* figscale, 6 \* figscale))

df.plot(kind='scatter', x=x\_colname, y=y\_colname, s=(32 \* figscale), alpha=alpha)

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = scatter\_plot(\_df\_4, \*['Pregnancies', 'Glucose'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def scatter\_plot(df, x\_colname, y\_colname, figscale=1, alpha=.8):

from matplotlib import pyplot as plt

plt.figure(figsize=(6 \* figscale, 6 \* figscale))

df.plot(kind='scatter', x=x\_colname, y=y\_colname, s=(32 \* figscale), alpha=alpha)

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = scatter\_plot(\_df\_5, \*['Glucose', 'BloodPressure'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def scatter\_plot(df, x\_colname, y\_colname, figscale=1, alpha=.8):

from matplotlib import pyplot as plt

plt.figure(figsize=(6 \* figscale, 6 \* figscale))

df.plot(kind='scatter', x=x\_colname, y=y\_colname, s=(32 \* figscale), alpha=alpha)

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = scatter\_plot(\_df\_6, \*['BloodPressure', 'SkinThickness'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def scatter\_plot(df, x\_colname, y\_colname, figscale=1, alpha=.8):

from matplotlib import pyplot as plt

plt.figure(figsize=(6 \* figscale, 6 \* figscale))

df.plot(kind='scatter', x=x\_colname, y=y\_colname, s=(32 \* figscale), alpha=alpha)

plt.gca().spines[['top', 'right',]].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = scatter\_plot(\_df\_7, \*['SkinThickness', 'Insulin'], \*\*{}) chart

<google.colab.\_quickchart\_helpers.SectionTitle at 0x7ab2d35b84c0> import numpy as np

from google.colab import autoviz

def value\_plot(df, y, figscale=1):

from matplotlib import pyplot as plt

df[y].plot(kind='line', figsize=(8 \* figscale, 4 \* figscale), title=y)

plt.gca().spines[['top', 'right']].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = value\_plot(\_df\_8, \*['Pregnancies'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def value\_plot(df, y, figscale=1):

from matplotlib import pyplot as plt

df[y].plot(kind='line', figsize=(8 \* figscale, 4 \* figscale), title=y)

plt.gca().spines[['top', 'right']].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = value\_plot(\_df\_9, \*['Glucose'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def value\_plot(df, y, figscale=1):

from matplotlib import pyplot as plt

df[y].plot(kind='line', figsize=(8 \* figscale, 4 \* figscale), title=y)

plt.gca().spines[['top', 'right']].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = value\_plot(\_df\_10, \*['BloodPressure'], \*\*{}) chart

import numpy as np

from google.colab import autoviz

def value\_plot(df, y, figscale=1):

from matplotlib import pyplot as plt

df[y].plot(kind='line', figsize=(8 \* figscale, 4 \* figscale), title=y)

plt.gca().spines[['top', 'right']].set\_visible(False)

plt.tight\_layout()

return autoviz.MplChart.from\_current\_mpl\_state()

chart = value\_plot(\_df\_11, \*['SkinThickness'], \*\*{}) chart

# Step 4: Data Preprocessing

*Data preprocessing is crucial to ensure the dataset is suitable for model development. Steps will include:*

* *Handling missing data through imputation.*
* *Normalizing and scaling numerical features.*
* *Encoding categorical variables.*
* *Balancing class distribution, as diabetes prediction datasets are often imbalanced.*

*# Handle missing values (if any)*

*# Example: Replace missing values in a specific column (e.g., Glucose) with the mean of that column* data['Glucose'].fillna(data['Glucose'].mean(), inplace=True)

*# Feature selection: Choose relevant features based on domain knowledge and data analysis*

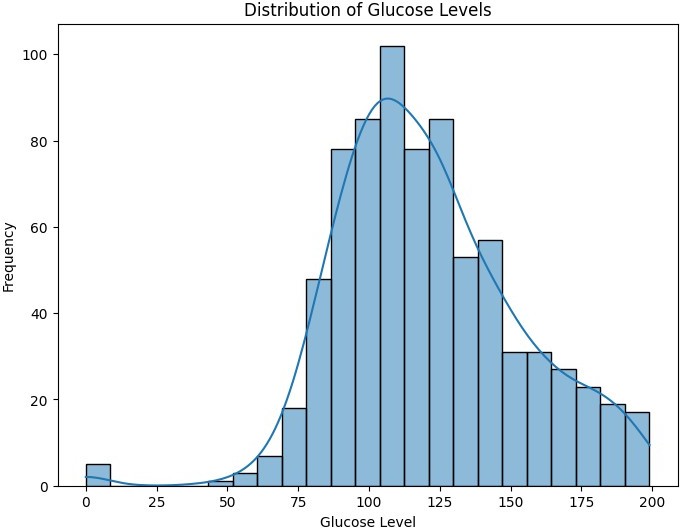
*# Example: Select relevant columns*

selected\_features = ['Glucose', 'BloodPressure', 'BMI', 'Age', 'Outcome']

data = data[selected\_features]

# Step 5: Data Visualization

*We'll create data visualizations to gain insights into the relationships between features and the distribution of diabetes cases. This will help us understand feature importance.*



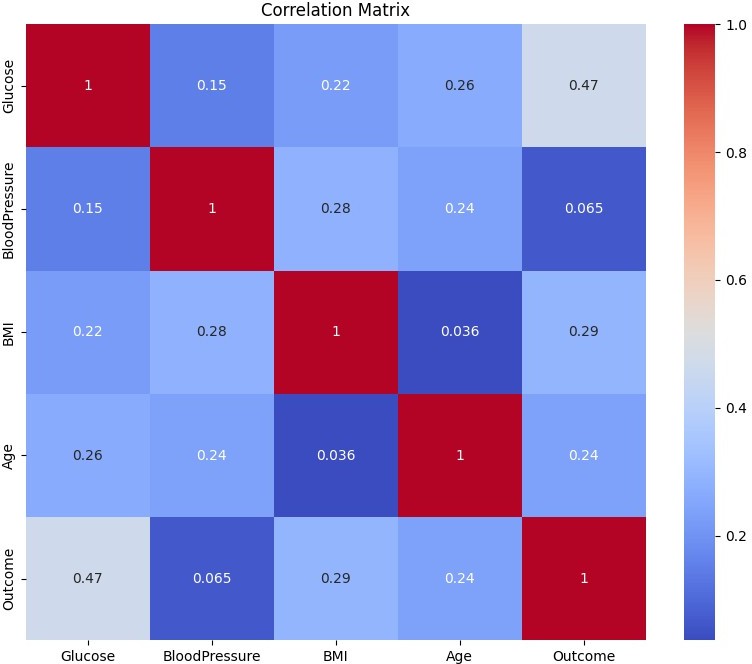
*# Visualize the distribution of glucose levels* plt.figure(figsize=(8, 6)) sns.histplot(data['Glucose'], kde=True) plt.title('Distribution of Glucose Levels') plt.xlabel('Glucose Level') plt.ylabel('Frequency')

plt.show()

*# Correlation matrix to see feature relationships* correlation\_matrix = data.corr() plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm') plt.title('Correlation Matrix')

plt.show()



# Step 6: Feature Engineering

*We may perform feature engineering to create new features or transform existing ones based on domain knowledge and insights from data visualization.*

*# Example: Creating a new feature for BMI categories*

def categorize\_bmi(bmi): if bmi < 18.5:

return 'Underweight' elif 18.5 <= bmi < 24.9:

return 'Normal' elif 25 <= bmi < 29.9:

return 'Overweight' else:

return 'Obese'

data['BMI Category'] = data['BMI'].apply(categorize\_bmi)

# Step 7: Data Splitting

*We will split the dataset into training and testing subsets to prepare for model development and evaluation.*

from sklearn.model\_selection import train\_test\_split

*# Define the features (X) and target (y)*

X = data.drop('Outcome', axis=1) y = data['Outcome']

*# Split the data into training and testing sets (e.g., 80% train, 20% test)*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

*# Check the shapes of the resulting sets* print("X\_train shape:", X\_train.shape) print("X\_test shape:", X\_test.shape)

X\_train shape: (614, 5)

X\_test shape: (154, 5)

Gayathri devi K

Nandha College Of Technology