DIABETES PREDICTION USING MACHINE LEARNING - spam classify

INTRODUCTION

Diabetes mellitus, a chronic metabolic disorder, has become a global health concern affecting millions of individuals worldwide. Timely diagnosis and intervention are paramount in managing this condition effectively. In this context, the integration of advanced machine learning techniques offers a promising avenue to improve the accuracy and efficiency of diabetes risk prediction.

ALGORITHM

Data Splitting:

Split the dataset into training and testing sets (e.g., 70% training, 30% testing) to assess model performance.

Model Selection:

Choose a suitable machine learning algorithm for classification tasks. Common choices include:

- Logistic Regression
- Decision Trees
- Random Forest
- Support Vector Machines
- Neural Networks

Model Evaluation:

Evaluate the model's performance using metrics like:

- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC

INPUT:

These features are used to make predictions about the likelihood of the individual having or developing diabetes.

1. Age: Age of the individual.
2. BMI (Body Mass Index): A measure of body fat based on height and weight.
3. Pregnancies.
4. Skin Thickness.
5. Insulin.
6. DiabetesPedigreeFunction.
7. Blood Pressure: Systolic and diastolic blood pressure values.
8. Glucose Levels: Fasting blood glucose levels.
9. Family History: Information about whether the individual has a family history of diabetes.
10. Physical Activity: Level of physical activity or exercise.
11. Diet: Dietary habits and nutritional information.
12. Outcome.

OUTPUT:

1. **Positive Outcome (1):** This indicates that the individual is at a higher risk of having or developing diabetes based on the input features.
2. **Negative Outcome (0):** This indicates that the individual is at a lower risk of having or developing diabetes based on the input features.

Project Methodology:

Our project will follow a systematic approach, beginning with data collection, preprocessing, and feature engineering. We will then explore various machine learning algorithms, including logistic regression, decision trees, ensemble methods, and potentially deep learning, to build and fine-tune predictive models.

To ensure the robustness of our model, we will employ cross-validation techniques and rigorous hyperparameter tuning. Additionally, we will focus on model explainability, using state-of-the-art methods to provide insights into the model's decision-making process.

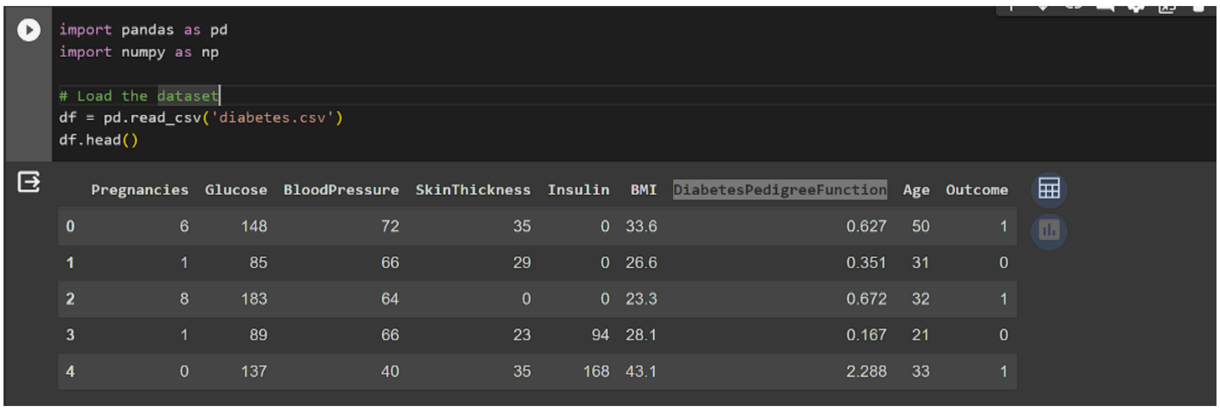
Expected Outcomes:

The successful completion of this project will result in a robust and interpretable machine learning model for diabetes risk prediction. This model can be integrated into healthcare systems, providing healthcare professionals with a valuable tool to identify at-risk individuals and tailor treatment plans accordingly.

Ultimately, our project aims to contribute to the advancement of healthcare by harnessing the potential of artificial intelligence to improve the lives of those affected by diabetes. By enhancing early diagnosis and personalized care, we aspire to reduce the impact of diabetes on individuals and communities.

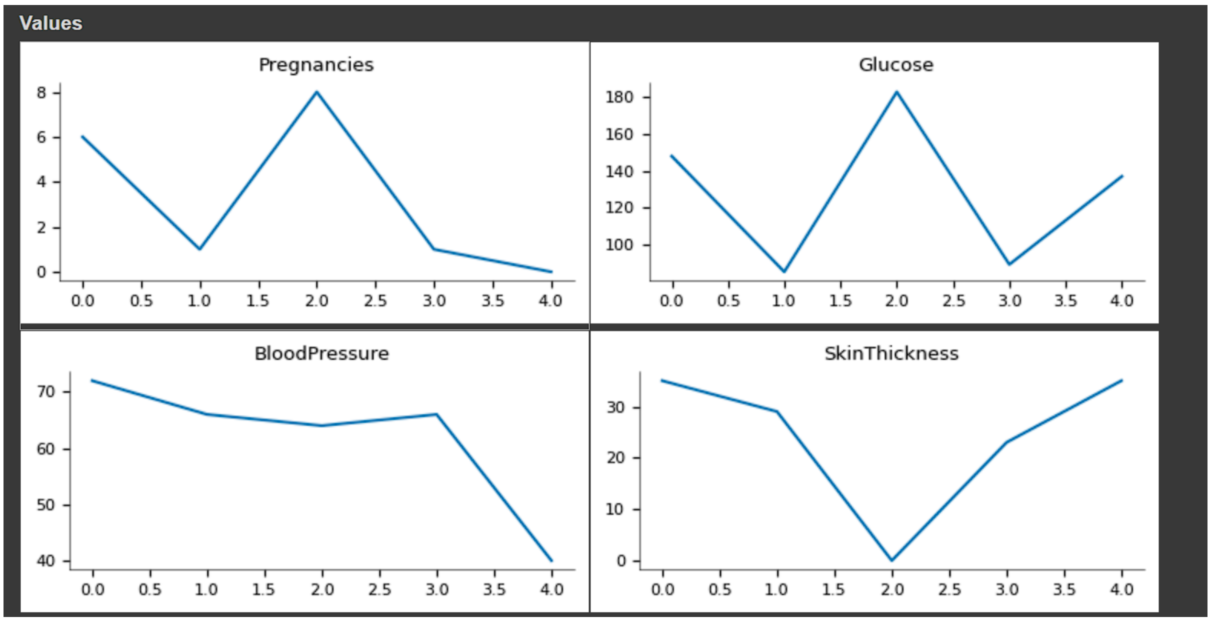
Data cleaning:

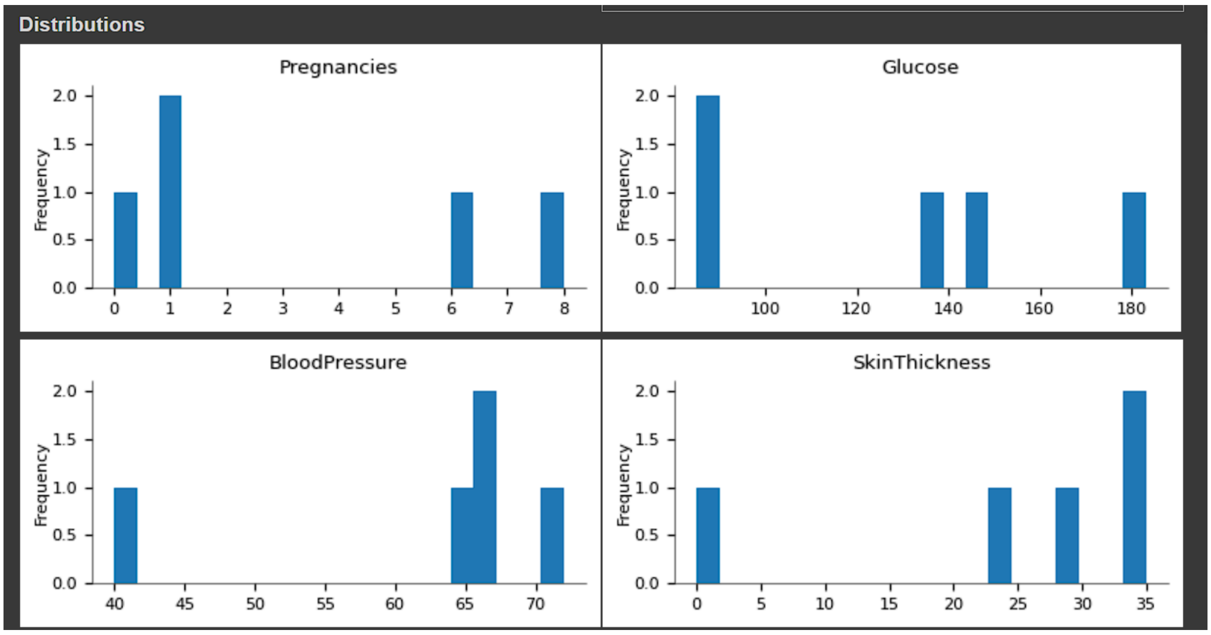
- Import the necessary libraries
- Load the dataset
- Check the data information using df.info()



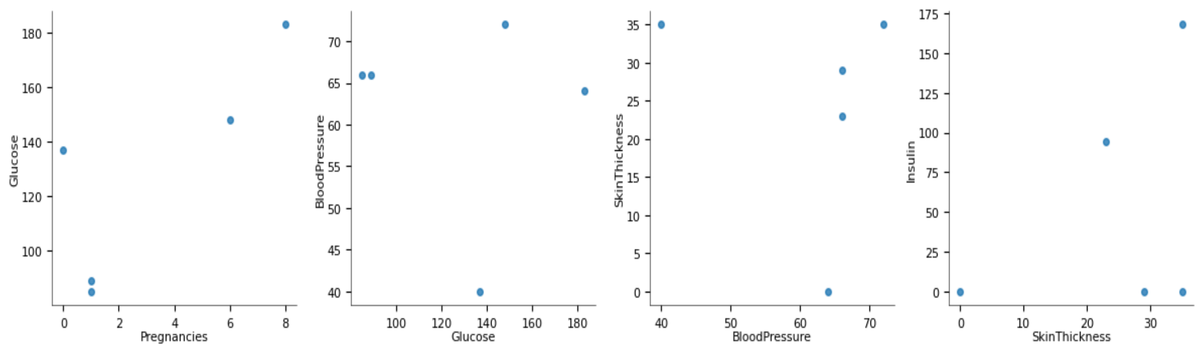
```python
import pandas as pd
import numpy as np

# Load the dataset
df = pd.read_csv('diabetes.csv')
df.head()
```

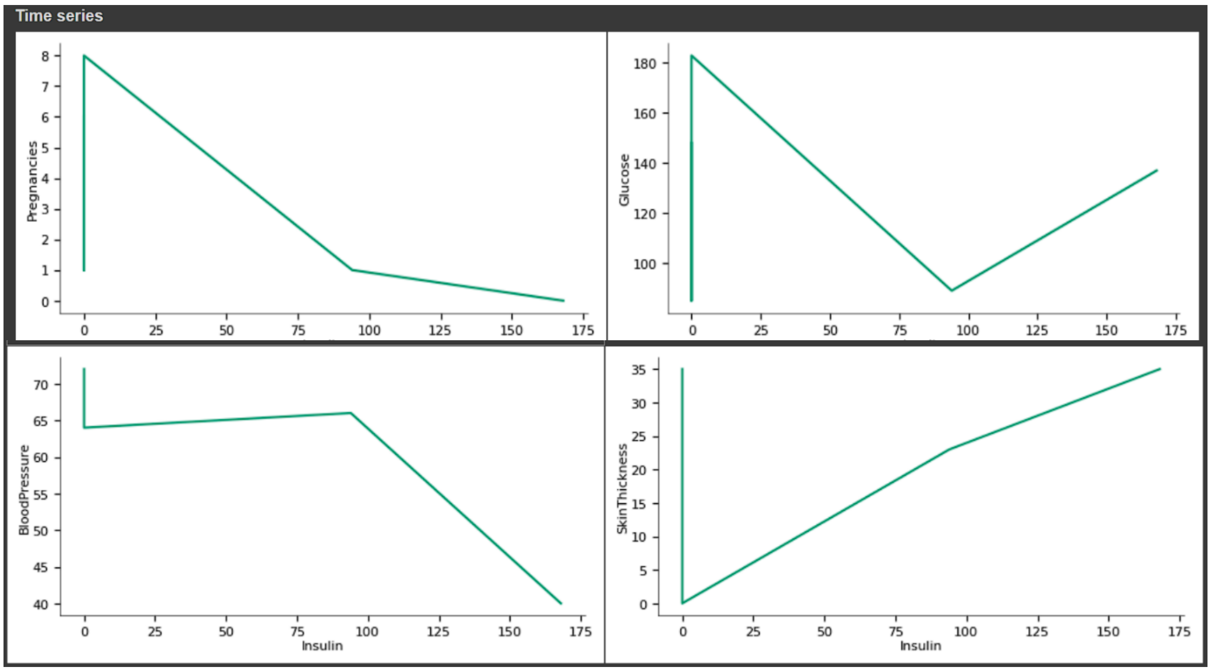
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72             35        0  33.6                     0.627   50        1
1            1       85             66             29        0  26.6                     0.351   31        0
2            8      183             64              0        0  23.3                     0.672   32        1
3            1       89             66             23       94  28.1                     0.167   21        0
4            0      137             40             35      168  43.1                     2.288   33        1
```





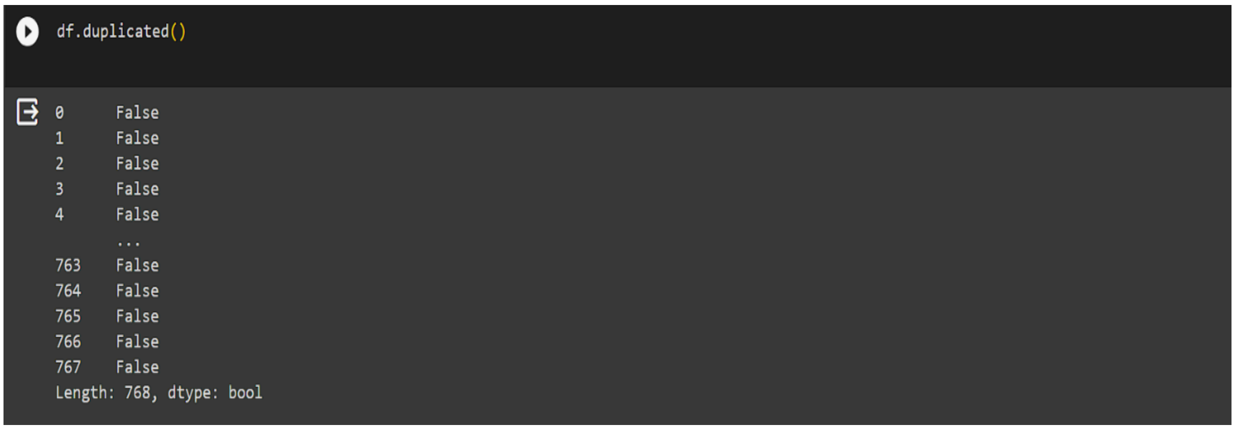
2-d distributions





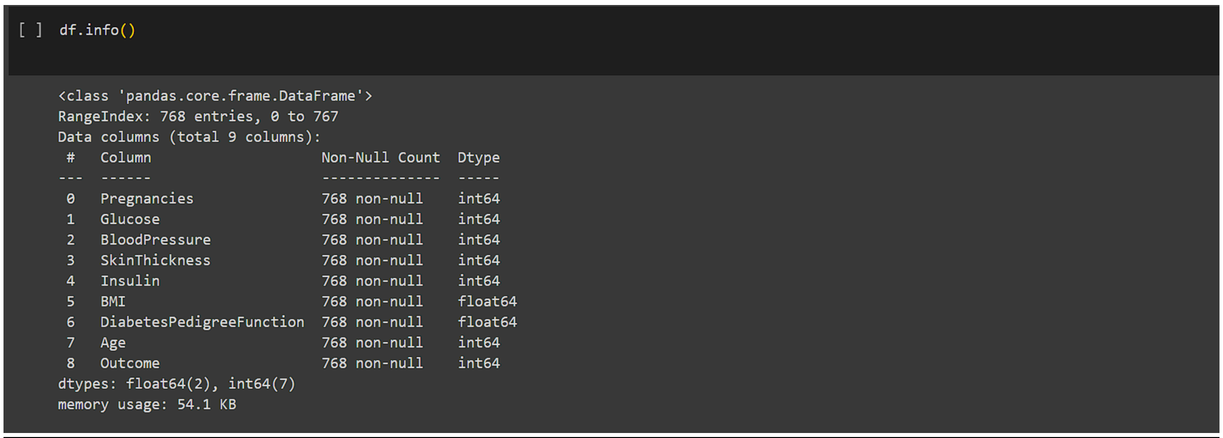
Data inspection and exploration:

=> Check the duplicate rows.



```
df.duplicated()
```

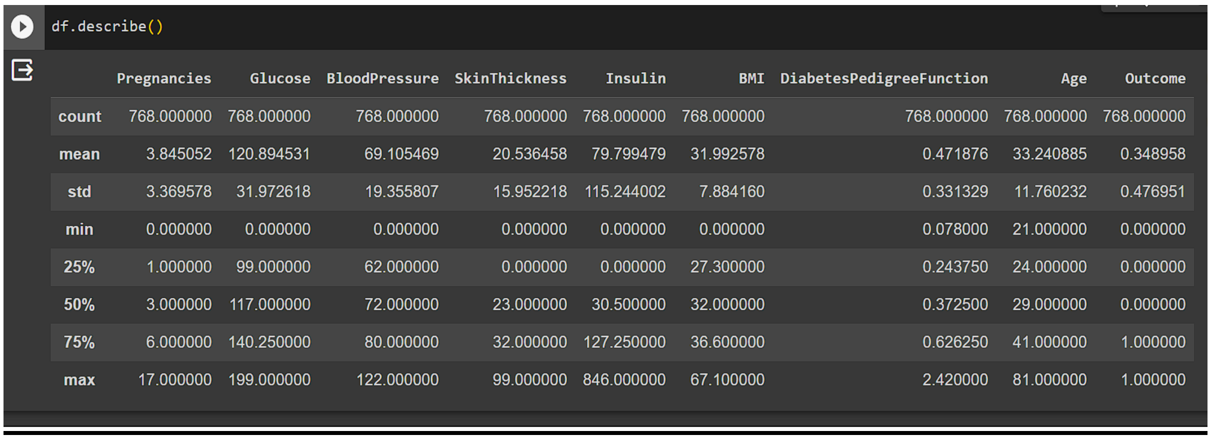
```
0      False
1      False
2      False
3      False
4      False
       ...
763    False
764    False
765    False
766    False
767    False
Length: 768, dtype: bool
```



```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

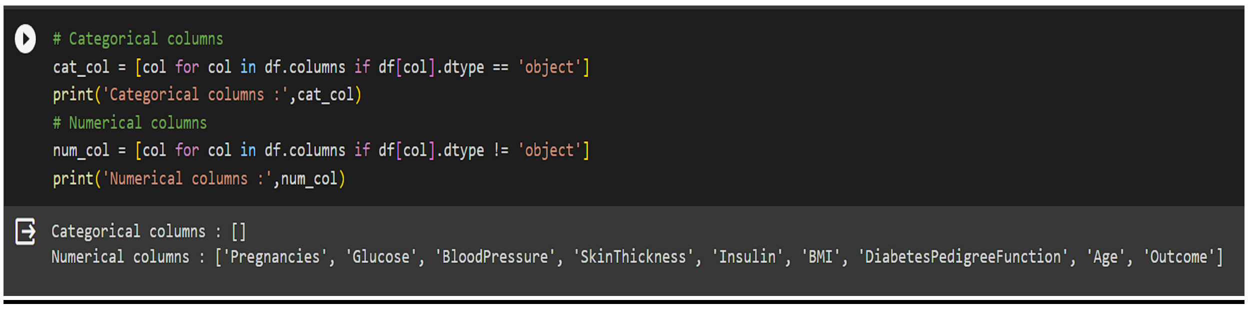
To Describe:



```
df.describe()
```

```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin         BMI  DiabetesPedigreeFunction         Age     Outcome
count   768.000000  768.000000     768.000000     768.000000  768.000000  768.000000                768.000000  768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479   31.992578                  0.471876   33.240885    0.348958
std       3.369578   31.972618      19.355807      15.952218  115.244002    7.884160                  0.331329   11.760232    0.476951
min       0.000000    0.000000       0.000000       0.000000    0.000000    0.000000                  0.078000   21.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000   27.300000                  0.243750   24.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000   32.000000                  0.372500   29.000000    0.000000
75%       6.000000  140.250000      80.000000      32.000000  127.250000   36.600000                  0.626250   41.000000    1.000000
max      17.000000  199.000000     122.000000      99.000000  846.000000   67.100000                  2.420000   81.000000    1.000000
```

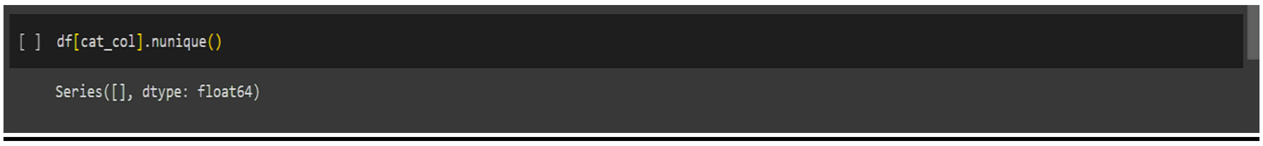
Check the categorical and numerical columns:



```
# Categorical columns
cat_col = [col for col in df.columns if df[col].dtype == 'object']
print('Categorical columns :',cat_col)
# Numerical columns
num_col = [col for col in df.columns if df[col].dtype != 'object']
print('Numerical columns :',num_col)
```

```
Categorical columns : []
Numerical columns : ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

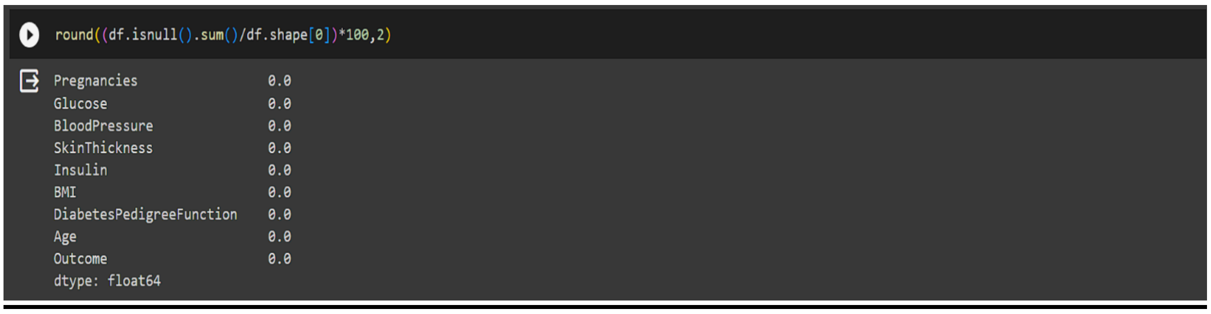
Check the total number of unique values in the Categorical columns:



```
[ ]  df[cat_col].nunique()
```

```
Series([], dtype: float64)
```

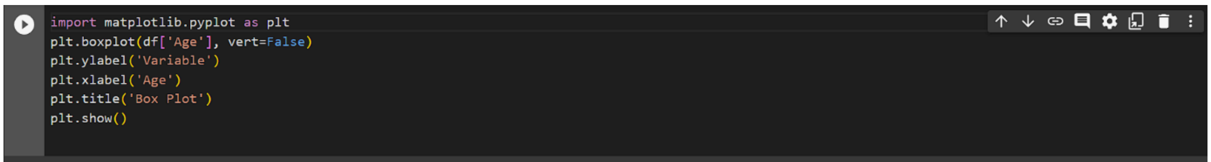
Handling missing data:



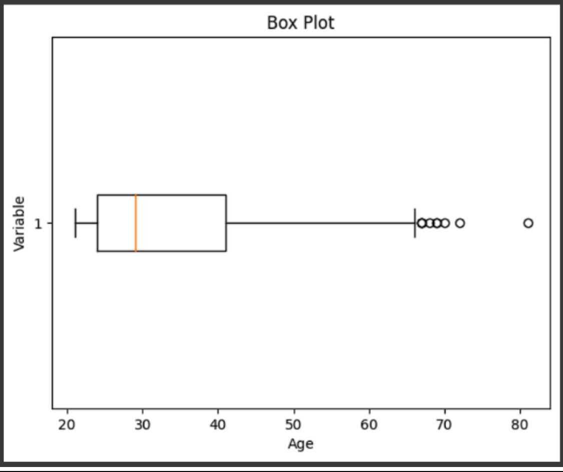
```
round((df.isnull().sum()/df.shape[0])*100,2)
```

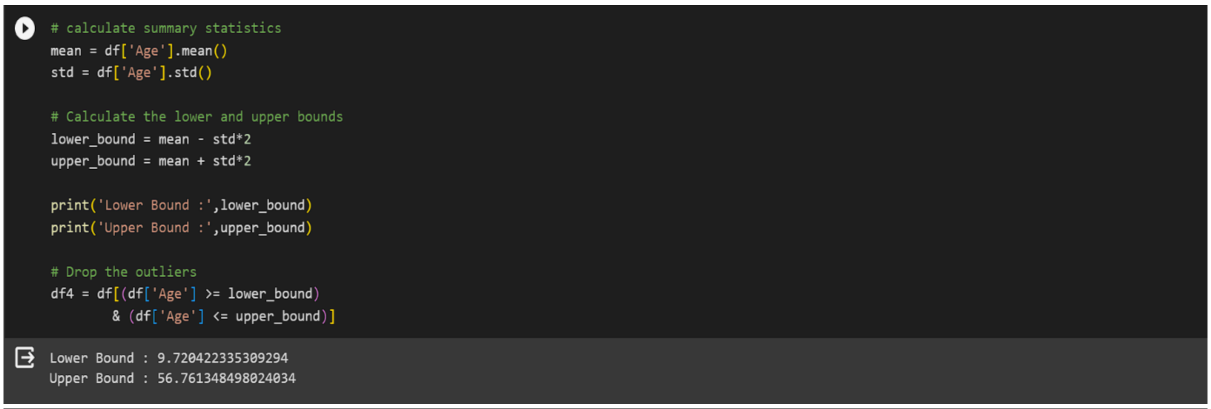
```
Pregnancies                 0.0
Glucose                     0.0
BloodPressure               0.0
SkinThickness               0.0
Insulin                     0.0
BMI                         0.0
DiabetesPedigreeFunction    0.0
Age                         0.0
Outcome                     0.0
dtype: float64
```

Handling outliers:



```
import matplotlib.pyplot as plt
plt.boxplot(df['Age'], vert=False)
plt.ylabel('Variable')
plt.xlabel('Age')
plt.title('Box Plot')
plt.show()
```

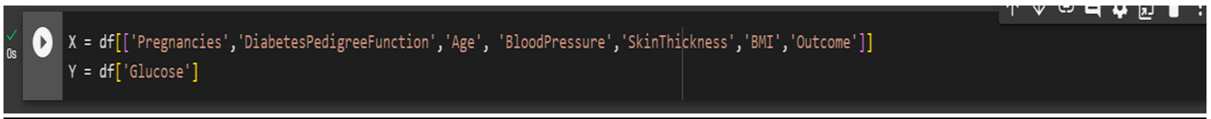




```
# calculate summary statistics
mean = df['Age'].mean()
std = df['Age'].std()

# Calculate the lower and upper bounds
lower_bound = mean - std*2
upper_bound = mean + std*2

print('Lower Bound :',lower_bound)
print('Upper Bound :',upper_bound)

# Drop the outliers
df4 = df[(df['Age'] >= lower_bound)
        & (df['Age'] <= upper_bound)]
```

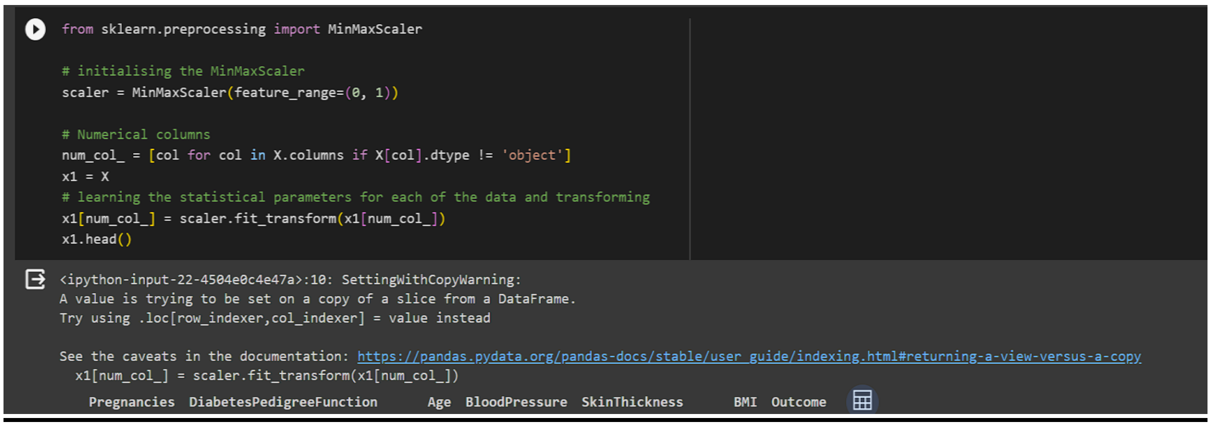
```
Lower Bound : 9.720422335309294
Upper Bound : 56.761348498024034
```

Data transformation



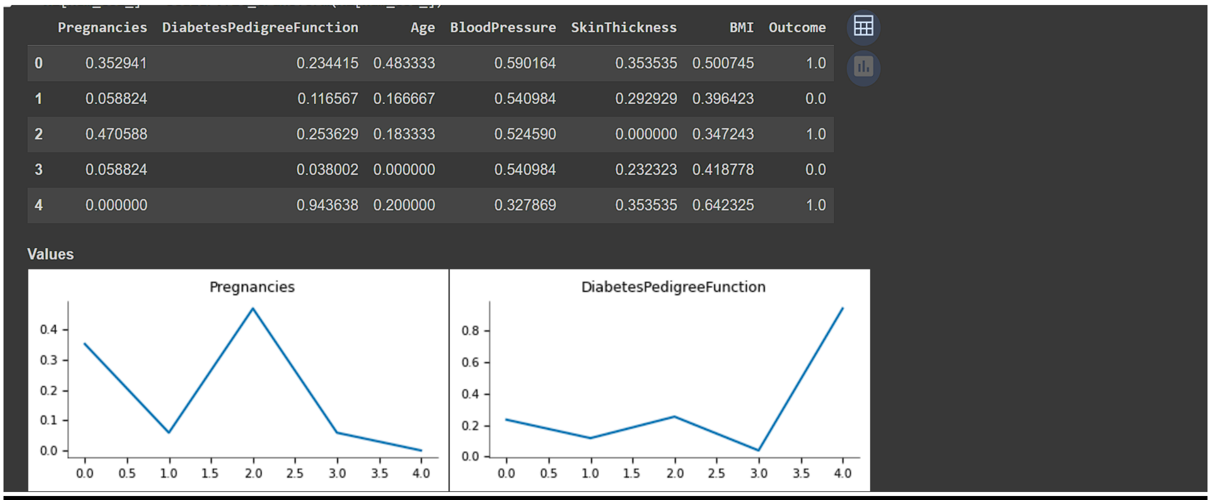
```
X = df[['Pregnancies','DiabetesPedigreeFunction','Age', 'BloodPressure','SkinThickness','BMI','Outcome']]
Y = df['Glucose']
```

Min-Max Scaling:

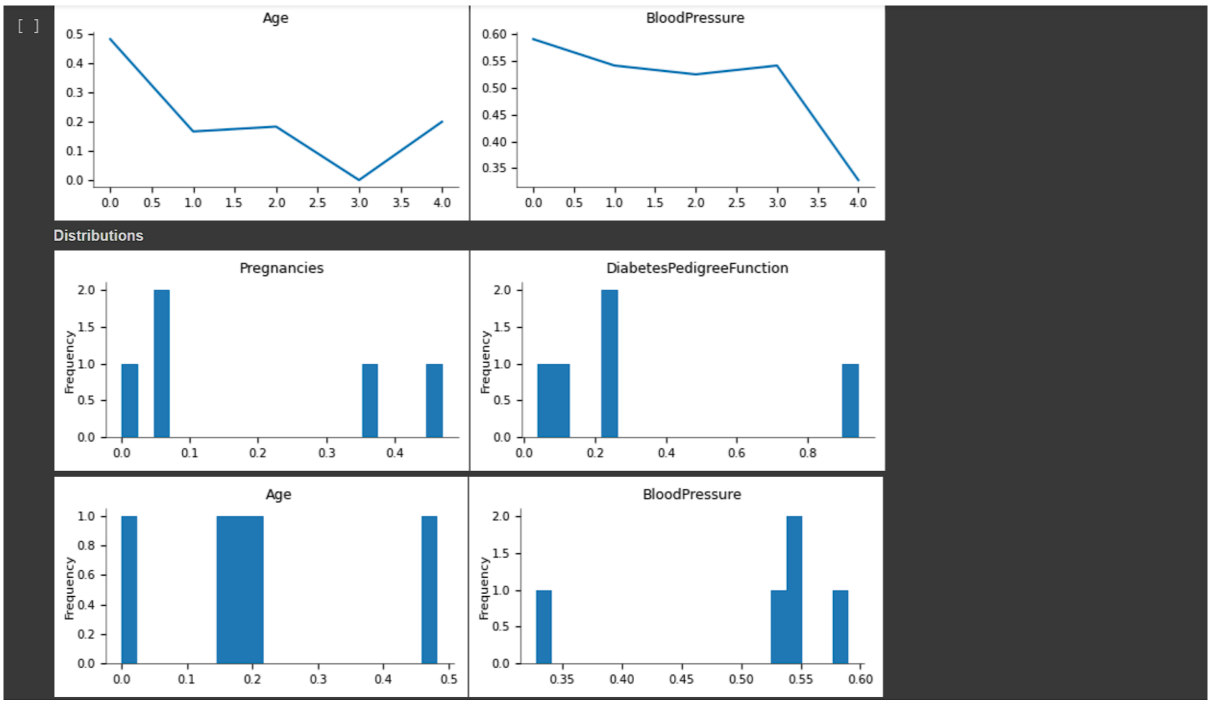


```
from sklearn.preprocessing import MinMaxScaler

# initialising the MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

# Numerical columns
num_col_ = [col for col in X.columns if X[col].dtype != 'object']
x1 = X
# learning the statistical parameters for each of the data and transforming
x1[num_col_] = scaler.fit_transform(x1[num_col_])
x1.head()
```

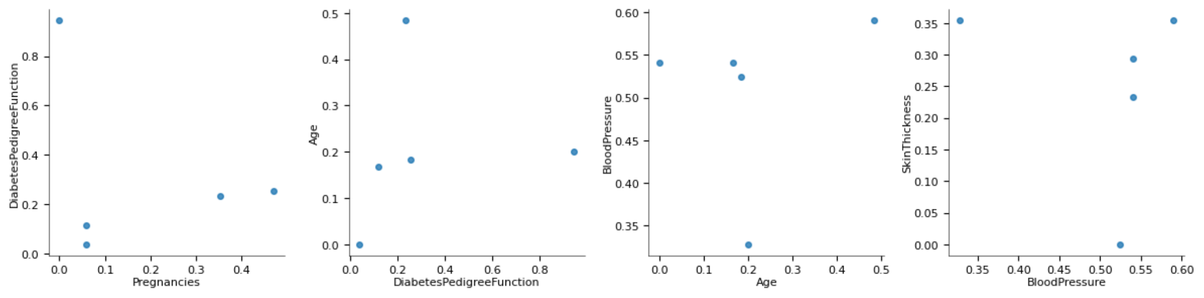
```
<ipython-input-22-4504e0c4e47a>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  x1[num_col_] = scaler.fit_transform(x1[num_col_])
    Pregnancies  DiabetesPedigreeFunction       Age  BloodPressure  SkinThickness       BMI  Outcome
```



	Pregnancies	DiabetesPedigreeFunction	Age	BloodPressure	SkinThickness	BMI	Outcome
0	0.352941	0.234415	0.483333	0.590164	0.353535	0.500745	1.0
1	0.058824	0.116567	0.166667	0.540984	0.292929	0.396423	0.0
2	0.470588	0.253629	0.183333	0.524590	0.000000	0.347243	1.0
3	0.058824	0.038002	0.000000	0.540984	0.232323	0.418778	0.0
4	0.000000	0.943638	0.200000	0.327869	0.353535	0.642325	1.0



2-d distributions



BUILDING THE SPAM CLASSIFIER ALGORITHM:

Building a spam classifier within a diabetes prediction system might not be a typical combination, as these tasks usually address different domains of data analysis. A spam classifier is used to identify and filter out unwanted or irrelevant messages, such as email or text messages. In contrast, diabetes prediction is focused on identifying individuals at risk of diabetes based on health-related data.

Spam Classification:

- Train or use a pre-trained spam classifier (commonly used for email or text messages) to identify and filter out spam or irrelevant text data.
- This classifier could be a machine learning model, rule-based system, or a combination of both, depending on your needs.

  ❖ It's important to note that integrating a spam classifier into a healthcare application involves data privacy and regulatory considerations. Ensure that you handle healthcare data responsibly and comply with relevant healthcare regulations such as HIPAA (in the United States) or GDPR (in Europe) to protect patient information.

ABSTRACT:

- The proposed framework consists of a multi-stage process, involving data collection, preprocessing, spam classification, and diabetes risk prediction. By implementing a spam classifier, we ensure that the healthcare data used for diabetes prediction is devoid of unwanted content, thus minimizing the risk of misinformation or data contamination.

- This innovative synergy of spam classification and predictive modeling not only streamlines data quality but also contributes to the overall robustness and ethical integrity of healthcare applications, thereby providing more accurate

and actionable insights for both healthcare professionals and patients. The results of this study underscore the potential benefits of such integrative approaches within the broader domain of healthcare data analytics and predictive modeling.

PSEUDOCODE:

```
# Data Collection
healthcare_data = load_healthcare_data()  # Load healthcare data, including text fields
# Data Preprocessing
processed_health_data = preprocess_healthcare_data(healthcare_data)  # Process health-related features
text_data = extract_text_data(healthcare_data)  # Extract text data
# Spam Classification
spam_classifier = load_spam_classifier_model()  # Load a pre-trained spam classifier
clean_text_data = remove_spam(text_data, spam_classifier)  # Remove spam or irrelevant text data
# Diabetes Prediction
diabetes_prediction_model = load_diabetes_prediction_model()  # Load the diabetes prediction model
predicted_diabetes_risk = predict_diabetes_risk(processed_health_data)  # Perform diabetes prediction
# Integration
integrated_data = merge_health_and_text_data(processed_health_data, clean_text_data)  # Merge processed data
final_result = combine_diabetes_and_spam_results(predicted_diabetes_risk, integrated_data)  # Combine diabetes prediction and spam results
# Output    return final_result
```

Need For Spam Classifier In Diabetes Prediction:

1. **Improved Data Quality**: A spam classifier helps to filter out irrelevant and potentially harmful text data. This ensures that only relevant and trustworthy healthcare data is used for diabetes prediction, improving the overall quality of the data.

2. **Reduced Noise**: Irrelevant text data, such as spam or unrelated comments, can introduce noise into the dataset, potentially leading to inaccurate predictions. Removing this noise through spam classification helps the diabetes prediction model focus on meaningful information.

3. **Enhanced Model Performance**: By providing cleaner and more relevant data, the diabetes prediction model is likely to perform better. It can make more accurate predictions, leading to improved patient risk assessments.

4. **Privacy and Security**: Spam classification can also help protect patient privacy and security. Spam messages often contain sensitive information, and by filtering them out, you reduce the risk of data breaches and unauthorized access.

5. **Ethical Considerations**: In healthcare, ethical considerations are crucial. Spam classification ensures that only ethically sourced and relevant data is used for predictions, aligning with healthcare regulations and best practices.

6. **Reduction in False Positives**: Filtering out spam and irrelevant data can help reduce false positives in diabetes prediction. Patients who may have been incorrectly identified as at risk due to irrelevant data are no longer impacted.

7. **Time and Resource Efficiency**: Spam classification reduces the need for healthcare professionals to manually review and clean the data, saving time and resources in the prediction process.

8. **Improved User Experience**: In healthcare applications, a clean and relevant dataset can lead to a better user experience for healthcare professionals and patients who interact with the system.

9. **Regulatory Compliance**: Integrating a spam classifier can help ensure that your diabetes prediction system complies with data privacy regulations, such as HIPAA, by minimizing the inclusion of irrelevant or unauthorized data.

10. **Continuous Monitoring**: Spam classifiers can be continuously updated and improved, enhancing their ability to adapt to evolving spam and irrelevant data sources, making the system more resilient and reliable over time.