Data preprocessing

Date	11 October 2023
Team id	Proj-212176-Team-2
Project Name	AI based Diabetes Prediction System
Maximum mark	

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining tasks.

Program :

Import the necessary libraries:

Numpy,pandas,sklearn ,matplotlib.pyplot

Explaination:

- Numpy :(import numpy as np) a library for mathematcal operatons and handling arrays.
- pandas :(import pandas as pd) a library for data manipulaton and analysis.
- Matplotlib.pyplot: (import as plt) a library for creatng visualiiaton.
- sklearn (preproccesing and evaluate model)

code:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler , Normalizer
from sklearn.compose import make_column_transformer, make_column_select
or from sklearn.model_selection import train_test_split
```

Import the dataset

dataset = pd.read_csv('C:/Users/91638/Documents/diabetes.csv')

Data preprocessing

Data preprocessing is a critical step in building an AI-based diabetes detection model. Properly processed data can significantly impact the performance of your model.

dataset.head()

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler , Normalizer
from sklearn.compose import make_column_transformer, make_column_selector
from sklearn.model_selection import train_test_split
```

In [3]:
```
dataset = pd.read_csv('C:/Users/91638/Documents/diabetes.csv')
dataset.head()
```

Out[3]:

	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

In [ ]:

dataset.info()

```
In [4]: dataset.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
```

preprocessor=make_column_transformer((StandardScaler(),make_column_selector(dtype_include=np.number)),)

preprocessor.fit(X)

X = preprocessor.transform(X)

```
In [11]: preprocessor = make_column_transformer(
             (StandardScaler(),
              make_column_selector(dtype_include=np.number)),
         )

In [12]: preprocessor.fit(X)
         X = preprocessor.transform(X)

In [13]: dataset.head()

Out[13]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
13	1	189.0	60.0	23.0	846.0	30.1	0.398	59	1

Data Cleaning:

- Handle missing values: Identify and handle missing data. You can either impute missing values or remove rows/columns with missing data depending on the extent of missingness.
- Outlier detection and treatment: Identify and deal with outliers in your data. Outliers can negatively impact model performance.

X = dataset.copy()

y = X.pop('Outcome')

```
In [8]: X = dataset.copy()
        y = X.pop('Outcome')

In [9]: dataset.head()
Out[9]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
13	1	189.0	60.0	23.0	846.0	30.1	0.398	59	1

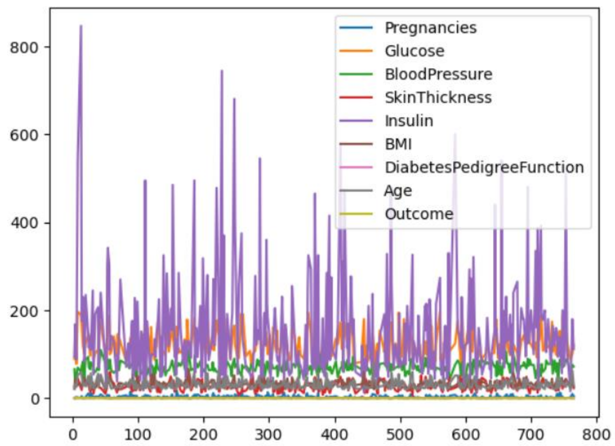
```
In [ ]:
```

This code demonstrates the basic steps for data cleaning, such as handling missing values, removing duplicates, and optionally addressing outliers, data types, column names, and reindexing. Adjust these steps as necessary based on your dataset's characteristics and the specific data quality issues you encounter.

Data visualization:

Data visualization is an important step in understanding your dataset when working on an AI-based diabetes detection project. You can use libraries like Matplotlib and Seaborn in Python to create various types of visualizations.

```
In [14]: dataset.plot()
Out[14]: <Axes: >
```

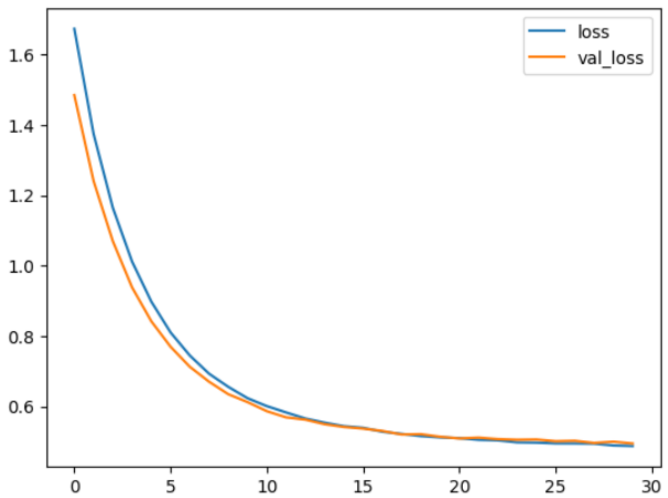


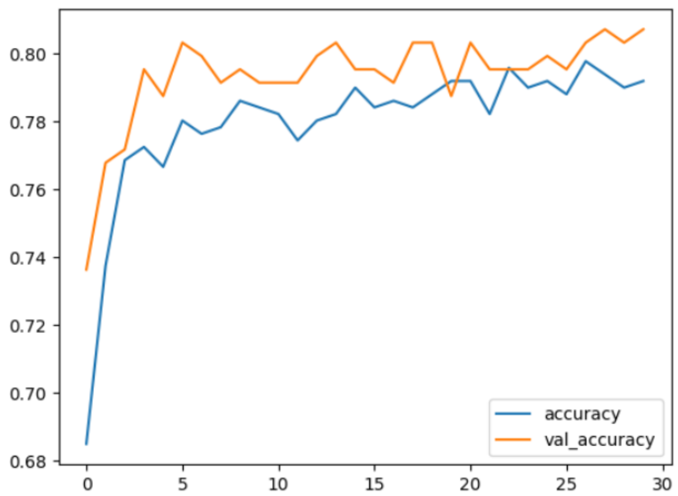
This code provides examples of various data visualization techniques:

1. Displaying the first few rows of the dataset to get an overview.
2. Generating summary statistics for numerical features.
3. Creating histograms to visualize the distribution of numerical features.
4. Generating boxplots to identify potential outliers.

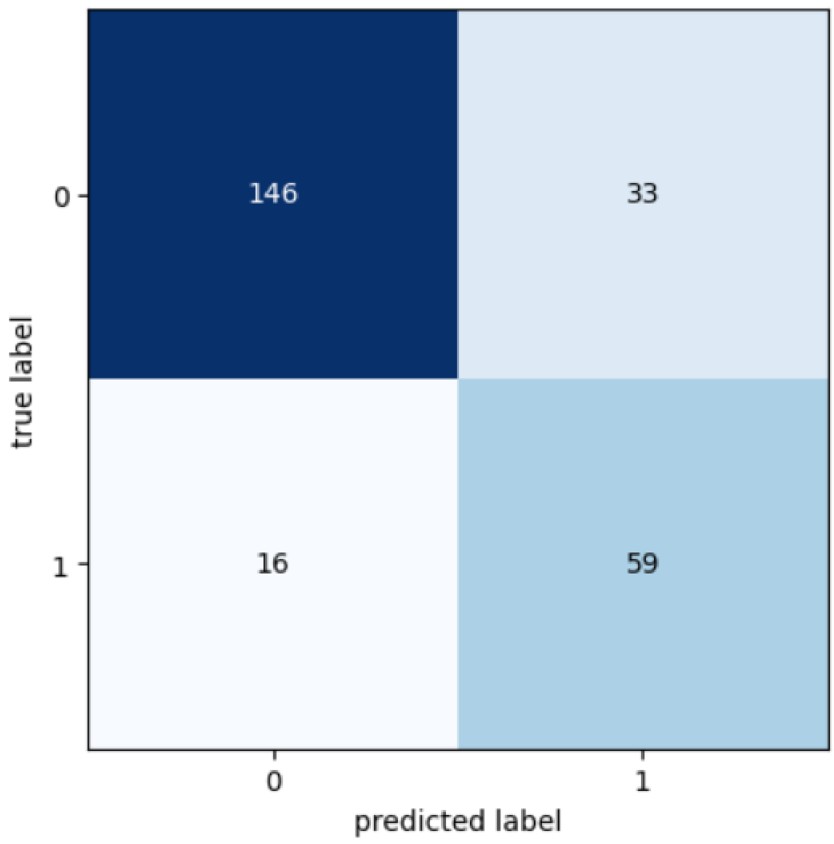
5. Creating a pairplot to visualize relationships between features, with hue indicating the outcome class.
6. Creating a correlation heatmap to visualize feature correlations.

```
history_df = pd.DataFrame(history.history)
history_df.loc[:, ['loss','val_loss']].plot();
history_df.loc[:, ['accuracy','val_accuracy']].plot();
```





```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
cm = confusion_matrix(y__predict, y__real)
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=cm)
plt.show()
```



Data Analysis:

Data analysis is a crucial step in developing an AI-based diabetes detection model. Through data analysis, you can gain insights into the dataset, understand the relationships between features, and make informed decisions about feature selection, preprocessing, and model development. Below are some key steps and code examples for data analysis in Python using popular libraries like Pandas, NumPy, and Matplotlib.

```
dataset.rename(columns={'DiabetesPedigreeFunction': 'DPF'}, inplace= True)
to_nan = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin']
to_nan.append(['BMI', 'DPF', 'Age'])
for i in range(len(to_nan)):
    dataset[to_nan[i]] = dataset[to_nan[i]].replace(0, np.nan)
dataset.head(10)
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DPF	Age	Outcome
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
13	1	189.0	60.0	23.0	846.0	30.1	0.398	59	1
14	5	166.0	72.0	19.0	175.0	25.8	0.587	51	1
16	0	118.0	84.0	47.0	230.0	45.8	0.551	31	1
18	1	103.0	30.0	38.0	83.0	43.3	0.183	33	0
19	1	115.0	70.0	30.0	96.0	34.6	0.529	32	1
20	3	126.0	88.0	41.0	235.0	39.3	0.704	27	0

dataset_true = dataset[(dataset.Outcome>0)]

dataset_true.describe().T

```
In [22]: dataset_true = dataset[(dataset.Outcome>0)]
         dataset_true.describe().T
```

Out[22]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	130.0	4.469231	3.916153	0.000	1.00000	3.000	7.0000	17.00
Glucose	130.0	145.192308	29.839388	78.000	124.25000	144.500	171.7500	198.00
BloodPressure	130.0	74.076923	13.021518	30.000	66.50000	74.000	82.0000	110.00
SkinThickness	130.0	32.961538	9.642770	7.000	26.00000	33.000	39.7500	63.00
Insulin	130.0	206.846154	132.699898	14.000	127.50000	169.500	239.2500	846.00
BMI	130.0	35.777692	6.734687	22.900	31.60000	34.600	38.3500	67.10
DPF	130.0	0.625585	0.405910	0.127	0.32975	0.546	0.7865	2.42
Age	130.0	35.938462	10.634705	21.000	27.25000	33.000	43.0000	60.00
Outcome	130.0	1.000000	0.000000	1.000	1.00000	1.000	1.0000	1.00

```
In [ ]:
```

dataset_false = dataset[(dataset.Outcome<1)]

dataset_false.describe().T

```
In [24]: dataset_false = dataset[(dataset.Outcome<1)]
         dataset_false.describe().T
```

Out[24]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	262.0	2.721374	2.617844	0.000	1.000	2.0000	4.00000	13.000
Glucose	262.0	111.431298	24.642133	56.000	94.000	107.5000	126.00000	197.000
BloodPressure	262.0	68.969466	11.892841	24.000	60.000	70.0000	76.00000	106.000
SkinThickness	262.0	27.251908	10.434135	7.000	18.250	27.0000	34.00000	60.000
Insulin	262.0	130.854962	102.626177	15.000	66.000	105.0000	163.75000	744.000
BMI	262.0	31.750763	6.794971	18.200	26.125	31.2500	36.10000	57.300
DPF	262.0	0.472168	0.299240	0.085	0.261	0.4135	0.62425	2.329
Age	262.0	28.347328	8.989008	21.000	22.000	25.0000	30.00000	81.000
Outcome	262.0	0.000000	0.000000	0.000	0.000	0.0000	0.00000	0.000

dataset.describe().T

In [25]:
```
dataset.describe().T
```

Out[25]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	392.0	3.301020	3.211424	0.000	1.00000	2.0000	5.000	17.00
Glucose	392.0	122.627551	30.860781	56.000	99.00000	119.0000	143.000	198.00
BloodPressure	392.0	70.663265	12.496092	24.000	62.00000	70.0000	78.000	110.00
SkinThickness	392.0	29.145408	10.516424	7.000	21.00000	29.0000	37.000	63.00
Insulin	392.0	156.056122	118.841690	14.000	76.75000	125.5000	190.000	846.00
BMI	392.0	33.086224	7.027659	18.200	28.40000	33.2000	37.100	67.10
DPF	392.0	0.523046	0.345488	0.085	0.26975	0.4495	0.687	2.42
Age	392.0	30.864796	10.200777	21.000	23.00000	27.0000	36.000	81.00
Outcome	392.0	0.331633	0.471401	0.000	0.00000	0.0000	1.000	1.00

In [ ]: