Experiment 01 - Preprocessing

Roll No.	19
Name	Manav Jawrani
Class	D15A
Subject	Business Intelligence Lab
LO Mapped	LO2: Organize and prepare the data needed for data mining algorithms in terms of attributes and class inputs, training, validating, and testing files.

Aim: To perform data preprocessing on the dataset using python.

Introduction:

1. What is Data Pre-processing?

Data preprocessing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for the data mining process. More recently, data preprocessing techniques have been adapted for training machine learning models and AI models and for running inferences against them.
Data preprocessing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the machine learning and AI development pipeline to ensure accurate results.
There are several different tools and methods used for preprocessing data, including the following:

- sampling, which selects a representative subset from a large population of data;
- transformation, which manipulates raw data to produce a single input;
- denoising, which removes noise from data;
- imputation, which synthesizes statistically relevant data for missing values;
- normalization, which organizes data for more efficient access; and
- feature extraction, which pulls out a relevant feature subset that is significant in a particular context.

These tools and methods can be used on a variety of data sources, including data stored in files or databases and streaming data.

2. Why is Data Preprocessing important?

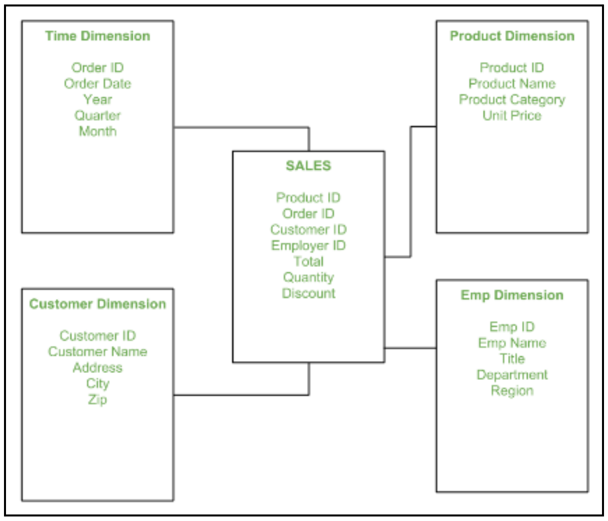
Virtually any type of data analysis, data science or AI development requires some type of data preprocessing to provide reliable, precise and robust results for enterprise applications.
Real-world data is messy and is often created, processed and stored by a variety of humans, business processes and applications. As a result, a data set may be missing individual fields, contain manual input errors, or have duplicate data or different names to describe the same thing. Humans can often identify and rectify these problems in the data they use in the line of business, but data used to train machine learning or deep learning algorithms needs to be automatically preprocessed.

3. Explain the dataset schema

A database schema is the logical representation of a database, which shows how the data is stored logically in the entire database. It contains a list of attributes and instructions that informs the database engine how the data is organized and how the elements are related to each other. A database schema contains schema objects that may include tables, fields, packages, views, relationships, primary key, foreign key. The schema does not physically contain the data itself; instead, it gives information about the shape of data and how it can be related to other tables or models.

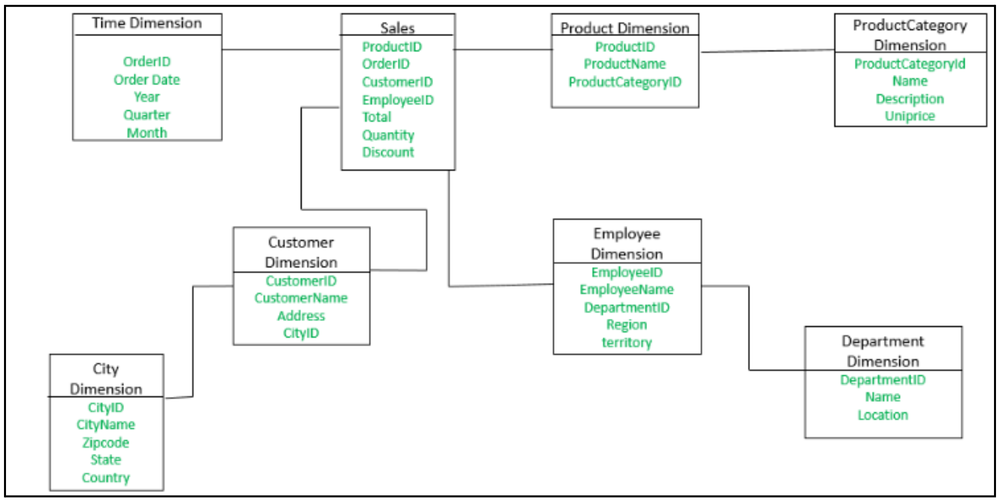
Star Schema

The star schema is a different way of schema design to organize the data. It is best suitable for storing and analyzing a huge amount of data, and it works on "Facts" and "Dimensions". Here the fact is the numerical data point that runs business processes, and Dimension is a description of fact. With Star Schema, we can structure the data of RDBMS.



Snowflake Schema

The snowflake schema is an adaptation of a star schema. There is a main "Fact" table in the star schema that contains the main data points and reference to its dimension tables. But in snowflakes, dimension tables can have their own dimension tables.



1. Data Cleaning

Real-world data tends to be incomplete, noisy, and inconsistent. Data Cleaning/Cleansing routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

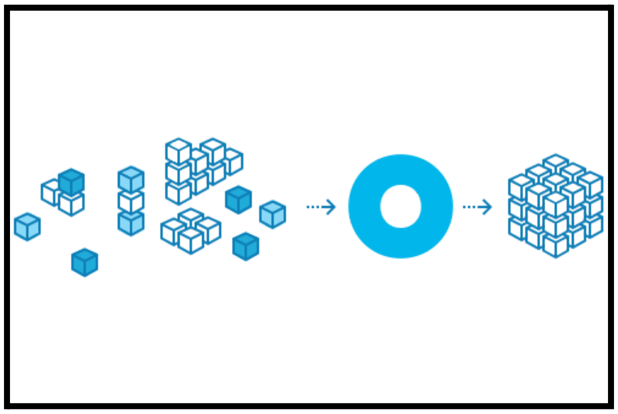
Data can be noisy, having incorrect attribute values. Owing to the following, the data collection instruments used may be at fault. Maybe human or computer errors occurred at data entry. Errors in data transmission can also occur.

“Dirty” data can cause confusion for the mining procedure. Although most mining routines have some procedures, they deal with incomplete or noisy data, which are not always robust. Therefore, a useful Data Preprocessing step is to run the data through some Data Cleaning/Cleansing routines.

2. Data Transformation

Data is transformed into appropriate forms of mining. Data Transformation involves the following:

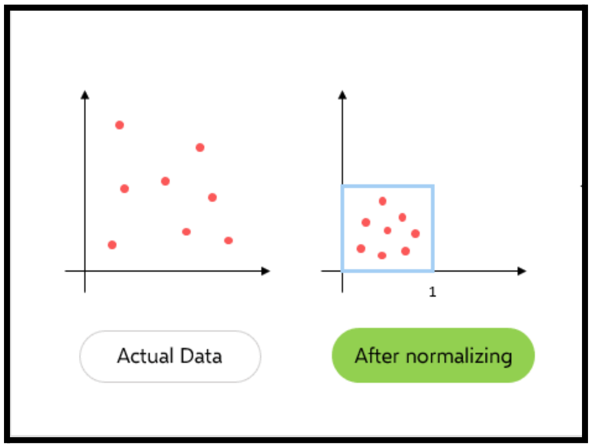
1. In Normalization, where the attribute data are scaled to fall within a small specified range, such as -1.0 to 1.0, or 0 to 1.0.
2. Smoothing works to remove the noise from the data. Such techniques include binning, clustering, and regression.
3. In Aggregation, summary or aggregation operations are applied to the data. For example, daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.
4. In Generalization of the Data, low level or primitive/raw data are replaced by higher level concepts through the use of concept hierarchies. For example, categorical attributes are generalized to higher level concepts street into city or country. Similarly, the values for numeric attributes may be mapped to higher level concepts like, age into young, middle-aged, or senior.



3. Data Normalization

Normalization is generally required when we are dealing with attributes on a different scale, otherwise, it may lead to a dilution in effectiveness of an equally important attribute(on a lower scale) because of other attributes having values on a larger scale.
In simple words, when multiple attributes are there but attributes have values on different scales, this may lead to poor data models while performing data mining operations. So they are normalized to bring all the attributes on the same scale.
The production of clean data is generally referred to as Data Normalization. However, when you dig a little deeper, the meaning or goal of Data Normalization is twofold:
Data Normalization is the process of organizing data such that it seems consistent across all records and fields.

It improves the cohesion of entry types, resulting in better data cleansing, lead creation, and segmentation.
Simply said, this procedure entails removing Unstructured Material as well as Redundancy (duplicates) to ensure logical data storage. When Data Normalization is done correctly, standardized data entry is the result. This technique, for example, applies to the recording of URLs, contact names, street locations, phone numbers, and even codes. These standardized data fields can thus be quickly grouped and read.

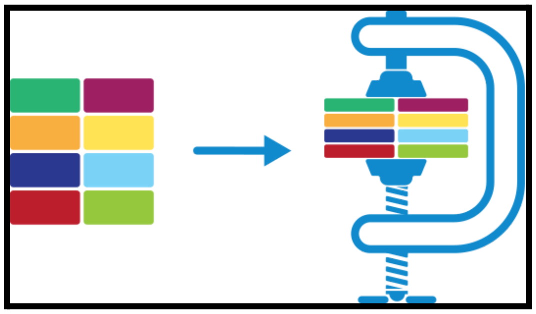


4. Data Reduction

Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data Reduction techniques are helpful in analyzing the reduced representation of the data set without compromising the integrity of the original data and yet producing the qualitative knowledge. Strategies for data reduction include the following:

1. In Data Cube Aggregation, aggregation operations are applied to the data in the construction of a data cube.
2. In Dimension Reduction, irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.
3. In Data Compression, encoding mechanisms are used to reduce data set size. The methods used for Data Compression are Wavelet Transform and Principal Component Analysis.
4. In Numerosity Reduction, data is replaced or estimated by alternative and smaller data representations such as parametric models (which store only the model parameters instead of the actual data, e.g. Regression and Log-Linear Models) or non-parametric methods (e.g. Clustering, Sampling, and the use of histograms).

5. In Discretization and Concept Hierarchy Generation, raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction and are powerful tools for data mining.



<u>Steps involved in Data Pre-processing</u>:

Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

1. **(a) Missing Data:**
   This situation arises when some data is missing in the data. It can be handled in various ways.
   Some of them are:
   1. **Ignore the tuples:**
      This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

   2. **Fill the Missing values:**
      There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

2. **(b). Noisy Data:**
   Noisy data is meaningless data that can’t be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :
   1. **Binning Method:**
      This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed

to complete the task. Each segment is handled separately.

2. **Regression:**
   Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

3. **Clustering:**
   This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

Data Transformation:
This step is taken in order to transform the data in appropriate forms suitable for the mining process. This involves following ways:

1. **Normalization:**
   It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. **Attribute Selection:**
   In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

3. **Discretization:**
   This is done to replace the raw values of numeric attributes by interval levels or conceptual levels.

4. **Concept Hierarchy Generation:**
   Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

Data Reduction:
Since data mining is a technique that is used to handle huge amounts of data. While working with a huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction techniques. It aims to increase the storage efficiency and reduce data storage and analysis costs.
The various steps to data reduction are:

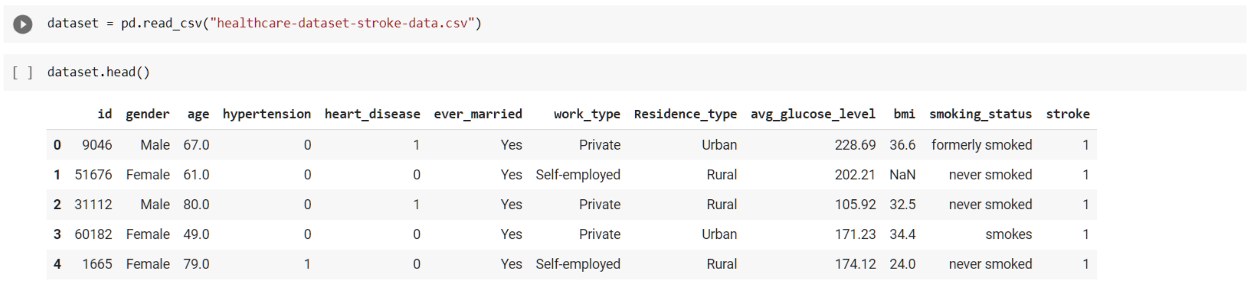
1. **Data Cube Aggregation:**
   Aggregation operation is applied to data for the construction of the data cube.

2. **Attribute Subset Selection:**
   The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use the level of significance and p- value of the attribute. The attribute having p-value greater than significance level can be discarded.

3. **Numerosity Reduction:**
   This enables us to store the model of data instead of whole data, for example: Regression Models.

4. **Dimensionality Reduction:**
   This reduces the size of data by encoding mechanisms. It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction is called lossless reduction, else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Component Analysis).

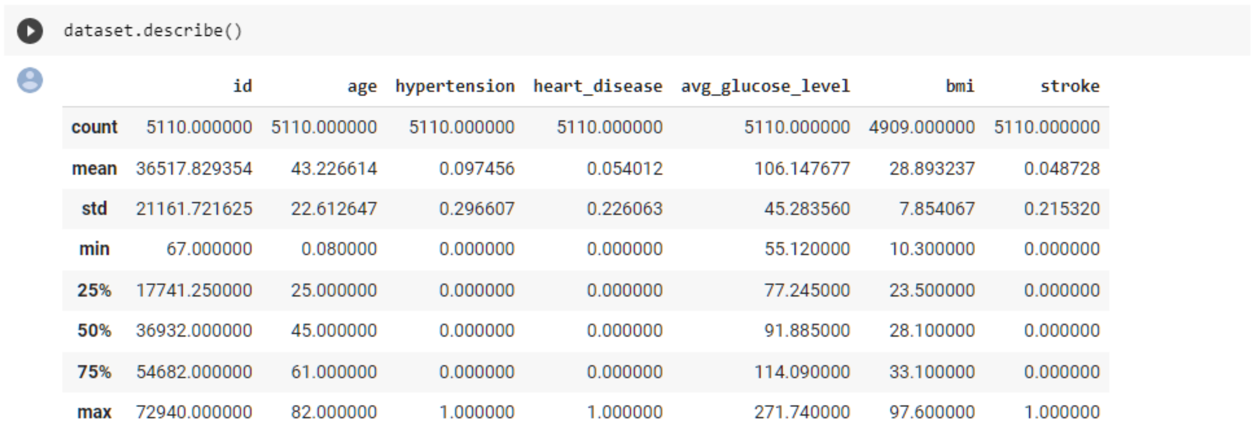
Results:



```
dataset = pd.read_csv("healthcare-dataset-stroke-data.csv")
```

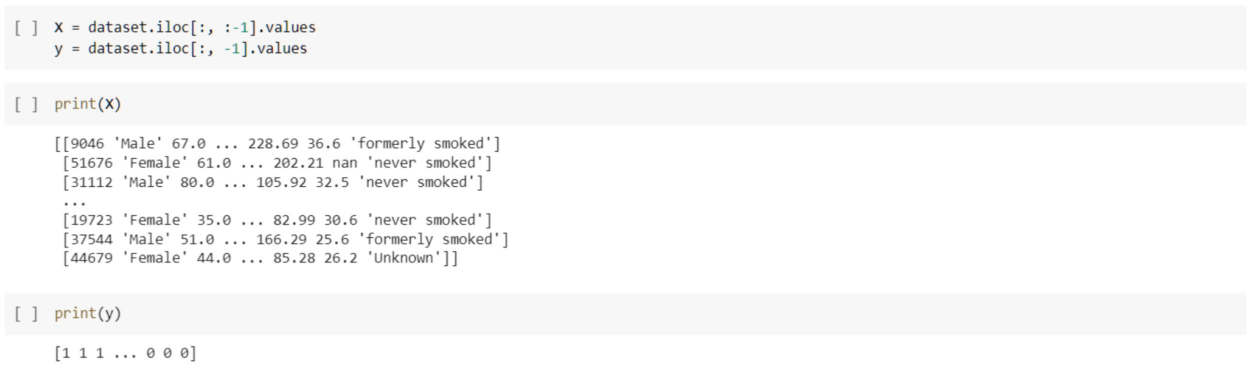
```
dataset.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1



```
dataset.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000



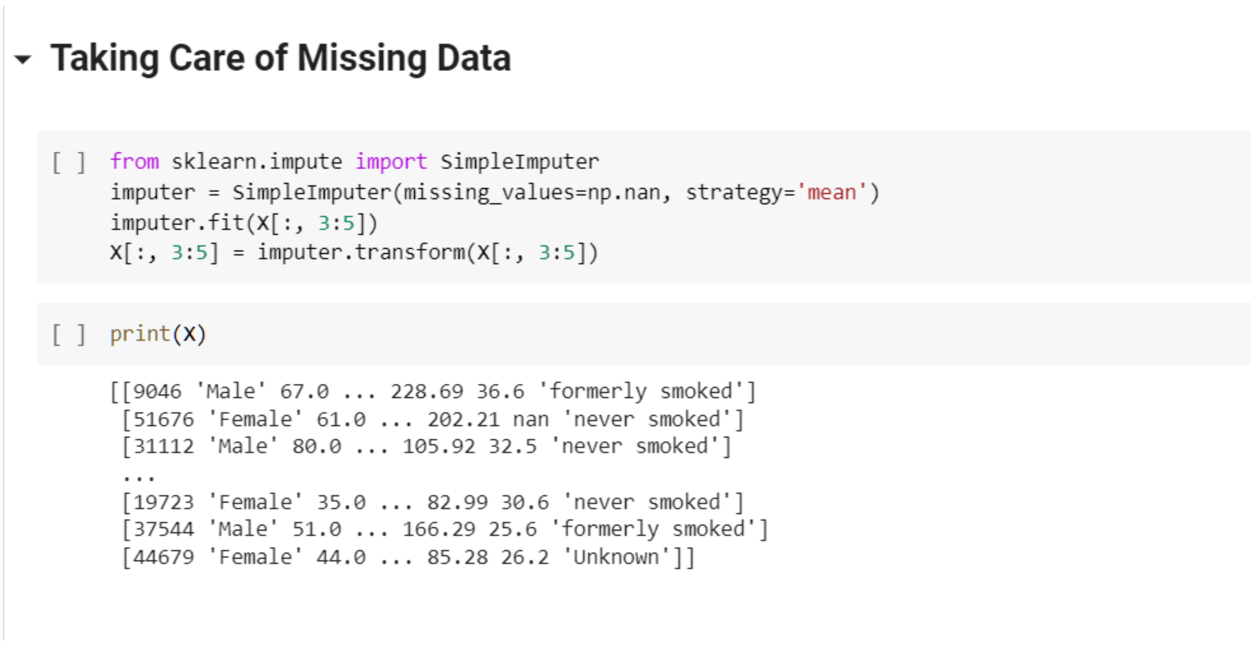
```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
print(X)
```

```
[[9046 'Male' 67.0 ... 228.69 36.6 'formerly smoked']
 [51676 'Female' 61.0 ... 202.21 nan 'never smoked']
 [31112 'Male' 80.0 ... 105.92 32.5 'never smoked']
 ...
 [19723 'Female' 35.0 ... 82.99 30.6 'never smoked']
 [37544 'Male' 51.0 ... 166.29 25.6 'formerly smoked']
 [44679 'Female' 44.0 ... 85.28 26.2 'Unknown']]
```

```
print(y)
```

```
[1 1 1 ... 0 0 0]
```

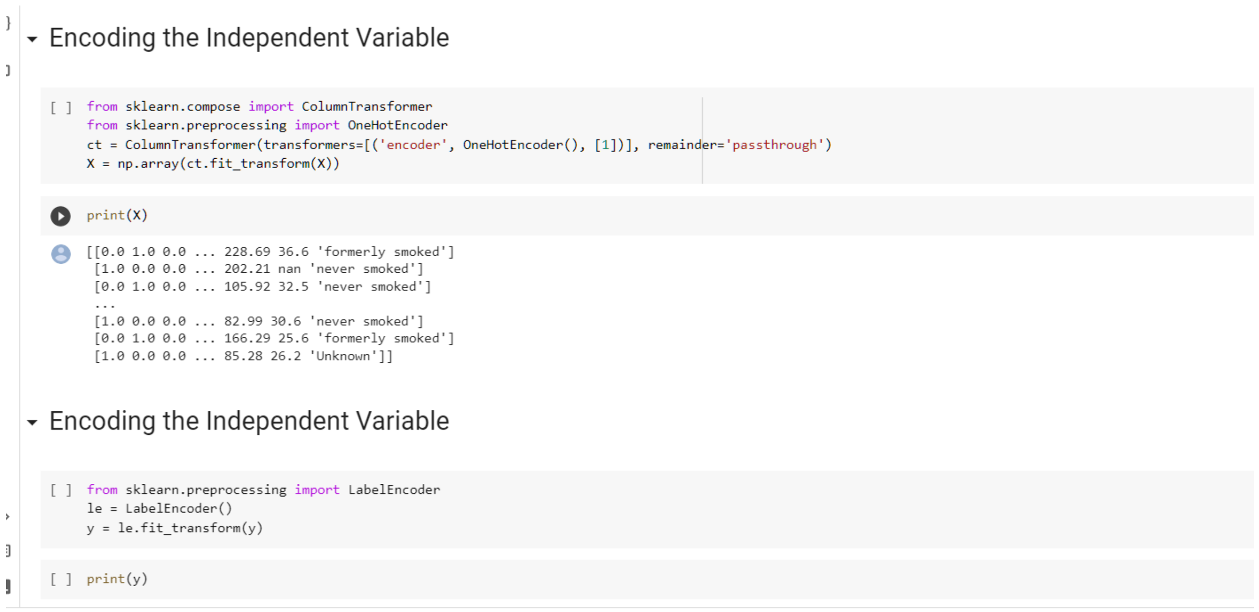


Taking Care of Missing Data

```
[ ] from sklearn.impute import SimpleImputer
    imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
    imputer.fit(X[:, 3:5])
    X[:, 3:5] = imputer.transform(X[:, 3:5])
```

```
[ ] print(X)
```

```
[[9046 'Male' 67.0 ... 228.69 36.6 'formerly smoked']
 [51676 'Female' 61.0 ... 202.21 nan 'never smoked']
 [31112 'Male' 80.0 ... 105.92 32.5 'never smoked']
 ...
 [19723 'Female' 35.0 ... 82.99 30.6 'never smoked']
 [37544 'Male' 51.0 ... 166.29 25.6 'formerly smoked']
 [44679 'Female' 44.0 ... 85.28 26.2 'Unknown']]
```



Encoding the Independent Variable

```
[ ] from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OneHotEncoder
    ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')
    X = np.array(ct.fit_transform(X))
```

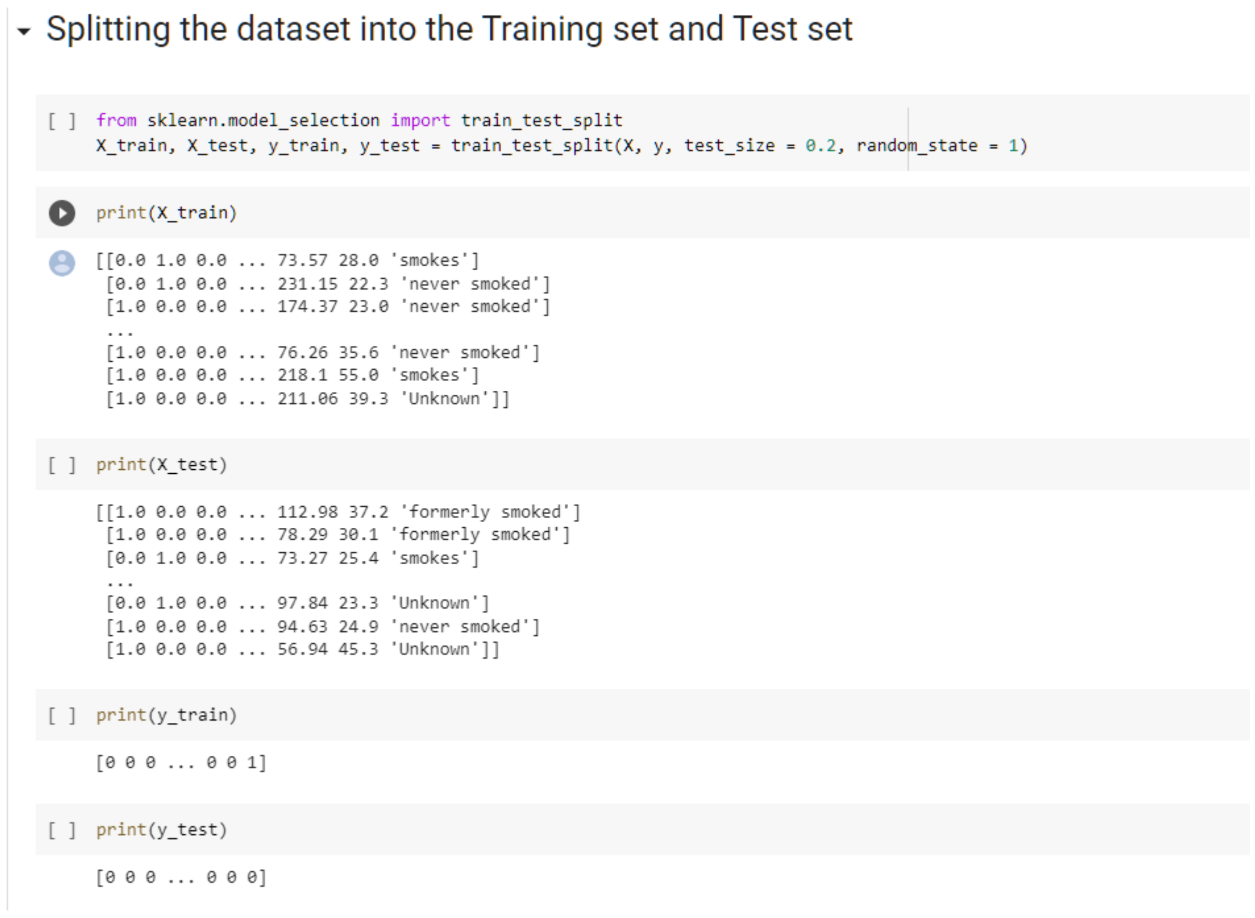
```
print(X)
```

```
[[0.0 1.0 0.0 ... 228.69 36.6 'formerly smoked']
 [1.0 0.0 0.0 ... 202.21 nan 'never smoked']
 [0.0 1.0 0.0 ... 105.92 32.5 'never smoked']
 ...
 [1.0 0.0 0.0 ... 82.99 30.6 'never smoked']
 [0.0 1.0 0.0 ... 166.29 25.6 'formerly smoked']
 [1.0 0.0 0.0 ... 85.28 26.2 'Unknown']]
```

Encoding the Independent Variable

```
[ ] from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    y = le.fit_transform(y)
```

```
[ ] print(y)
```



Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```

```
print(X_train)
```

```
[[0.0 1.0 0.0 ... 73.57 28.0 'smokes']
 [0.0 1.0 0.0 ... 231.15 22.3 'never smoked']
 [1.0 0.0 0.0 ... 174.37 23.0 'never smoked']
 ...
 [1.0 0.0 0.0 ... 76.26 35.6 'never smoked']
 [1.0 0.0 0.0 ... 218.1 55.0 'smokes']
 [1.0 0.0 0.0 ... 211.06 39.3 'Unknown']]
```

```
print(X_test)
```

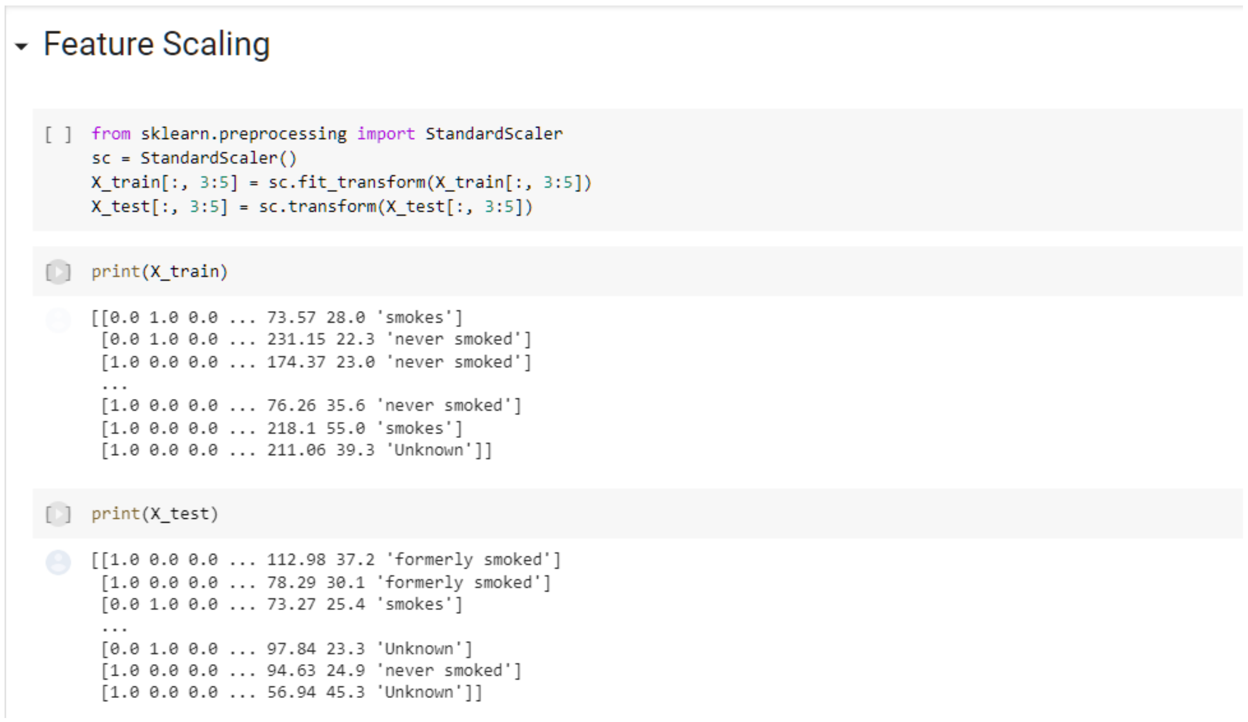
```
[[1.0 0.0 0.0 ... 112.98 37.2 'formerly smoked']
 [1.0 0.0 0.0 ... 78.29 30.1 'formerly smoked']
 [0.0 1.0 0.0 ... 73.27 25.4 'smokes']
 ...
 [0.0 1.0 0.0 ... 97.84 23.3 'Unknown']
 [1.0 0.0 0.0 ... 94.63 24.9 'never smoked']
 [1.0 0.0 0.0 ... 56.94 45.3 'Unknown']]
```

```
print(y_train)
```

```
[0 0 0 ... 0 0 1]
```

```
print(y_test)
```

```
[0 0 0 ... 0 0 0]
```



Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, 3:5] = sc.fit_transform(X_train[:, 3:5])
X_test[:, 3:5] = sc.transform(X_test[:, 3:5])
```

```
print(X_train)
```

```
[[0.0 1.0 0.0 ... 73.57 28.0 'smokes']
 [0.0 1.0 0.0 ... 231.15 22.3 'never smoked']
 [1.0 0.0 0.0 ... 174.37 23.0 'never smoked']
 ...
 [1.0 0.0 0.0 ... 76.26 35.6 'never smoked']
 [1.0 0.0 0.0 ... 218.1 55.0 'smokes']
 [1.0 0.0 0.0 ... 211.06 39.3 'Unknown']]
```

```
print(X_test)
```

```
[[1.0 0.0 0.0 ... 112.98 37.2 'formerly smoked']
 [1.0 0.0 0.0 ... 78.29 30.1 'formerly smoked']
 [0.0 1.0 0.0 ... 73.27 25.4 'smokes']
 ...
 [0.0 1.0 0.0 ... 97.84 23.3 'Unknown']
 [1.0 0.0 0.0 ... 94.63 24.9 'never smoked']
 [1.0 0.0 0.0 ... 56.94 45.3 'Unknown']]
```

Conclusion:
Hence we have successfully performed Pre-processing on the Dataset of **“Stroke Prediction”.**