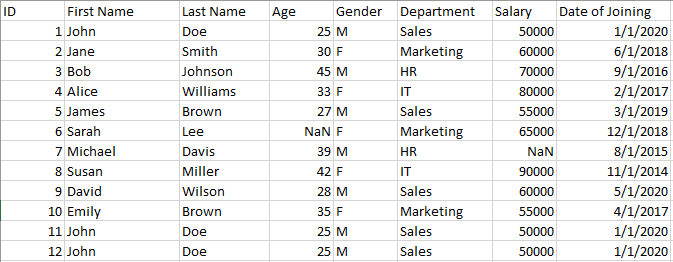
**“*bad data leads to bad predictive models*”**

***data.cvs*** dataset shown below:

  
data.csv showing various imperfections such as duplicated data, NaN, etc. Data created by Author.

**Libraries For Data Cleaning in Python**

In Python, a range of libraries and tools, including pandas and NumPy, may be used to clean up data. For instance, the *dropna()*, *drop duplicates()*, and *fillna()* functions in pandas may be used to manage missing data, remove missing data, and remove duplicate rows, respectively. The scikit-learn toolkit offers tools for dealing with outliers (such as the *SimpleImputer* class) and transforming data into a format that can be utilized by a model, such as the *StandardScaler* class for standardizing normalizing numerical data, and the *MinMaxScalar* for normalizing data.

In this article, we will explore various data cleaning techniques that can be used in Python to prepare and preprocess data for use in a machine learning model.

**Processing Missing Data**

The processing of missing data is one of the most important imperfections in a dataset. Several methods for dealing with missing data are provided by the pandas package in Python, including **dropna()** and **fillna().**The**dropna()** method is used to eliminate any columns or rows that have missing values. For instance, the code below will eliminate all rows with at least one missing value:

import pandas as pd

data = pd.read\_csv('data.csv')

data = data.dropna()

The **fillna()** function can be used to fill in missing values with a specific value or method. For example, the following code will fill in missing values in the 'age' column with the mean age of the data:

import pandas as pd

data = pd.read\_csv('data.csv')

data['age'].fillna(data['age'].mean(), inplace=True)

**Handling Outliers**

Handling outliers is a typical data cleaning activity. Values that diverge greatly from the rest of the data are considered outliers. These factors should be managed carefully since they have a significant influence on a model's performance. The *RobustScaler* class from the scikit-learn toolkit in Python is used to handle outliers. By deleting the median and scaling the data according to the interquartile range, this class may be used to scale the data.

from sklearn.preprocessing import RobustScaler

data = pd.read\_csv('data.csv')

scaler = RobustScaler()

data = scaler.fit\_transform(data)

**Encoding Categorical Variables**

Another common data cleaning task is converting data into a format that can be used by a model. For instance, before categorical data can be employed in a model, it must be transformed into numerical data. The *get\_dummies()* method in the pandas package allows one to transform category data into numerical data. In the example below, the categorical feature *‘Departmen*t’ is transformed into numerical data:

import pandas as pd

data = pd.read\_csv('data.csv')

data = pd.get\_dummies(data, columns=['Department'])

**Removing Duplicate Data**

Duplicate data must also be eliminated during the data cleaning process. To delete duplicate rows from a Python DataFrame, the *drop\_duplicates()* method provided by the pandas package can be used. For instance, the code below will eliminate any redundant rows from the data:

import pandas as pd

data = pd.read\_csv('data.csv')

data = data.drop\_duplicates()

# Feature Engineering

Feature selection and feature engineering are essential components of data cleaning. The process of choosing only the relevant features in a dataset is referred to as feature selection, whereas the process of building new features from already existing ones is known as feature engineering.  The code below is an illustration of feature engineering:

import pandas as pd

from sklearn.preprocessing import StandardScaler

# read the data into a pandas dataframe

df = pd.read\_excel("data.csv")

# create a feature matrix and target vector

X = df.drop(["Employee ID", "Date of Joining"], axis=1)

y = df["Salary"]

# scale the numerical features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X[["Age", "Experience"]])

# concatenate the scaled features with the categorical features

gender\_dummies = pd.get\_dummies(X["Gender"], prefix="Gender")

X\_processed = pd.concat(

[gender\_dummies, pd.DataFrame(X\_scaled, columns=["Age", "Experience"])],

axis=1,

)

print(X\_processed)

In the above code, we first create a feature matrix (X) by dropping the '*Employee ID*' and '*Date of Joining*' columns, and create a target vector (y) consisting of the '*Salary*' column. We then scale the numerical features '*Age*' and '*Experience*' using the StandardScaler() function from scikit-learn.

Next, we create dummy variables for the categorical '*Gender*' column and concatenate them with the scaled numerical features to create the final processed feature matrix (X\_processed).

Note that the specific feature extraction techniques used will depend on the data and the specific requirements of the analysis. Also, it's important to split the data into training and testing sets before applying any machine learning models to avoid overfitting.