Data Mining

Project 02: Data Preprocessing

Ramy Othman

About Dataset

- The venerable insurance industry is no stranger to data driven decision making. Yet in today's rapidly transforming digital landscape, Insurance is struggling to adapt and benefit from new technologies compared to other industries, even within the BFSI sphere (compared to the Banking sector for example.) Extremely complex underwriting rule-sets that are radically different in different product lines, many non-KYC environments with a lack of centralized customer information base, complex relationship with consumers in traditional risk underwriting where sometimes customer centricity runs reverse to business profit, inertia of regulatory compliance - are some of the unique challenges faced by Insurance Business.

About Dataset cont.

- This dataset can be helpful in a simple yet illuminating study in understanding the risk underwriting in Health Insurance, the interplay of various attributes of the insured and see how they affect the insurance premium.

Content

- This dataset contains 1338 rows of insured data, where the Insurance charges are given against the following attributes of the insured: Age, Sex, BMI, Number of Children, Smoker and Region. There are no missing or undefined values in the dataset.

Data Preprocessing

- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation and Data Discretization

Importing the required libraries

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
0]: from sklearn.decomposition import PCA
```

The imported libraries are:

1. Pandas
2. Numpy
3. Matplotlib
4. Seaborn
5. Sklearn

Read the dataset

```
In [2]: ds = pd.read_csv('insurance.csv')
        ds
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

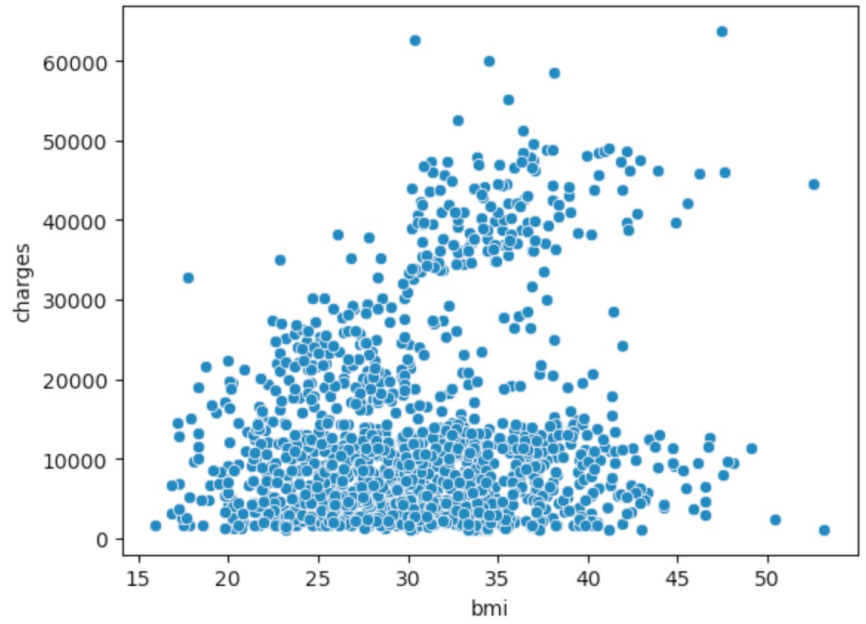
1338 rows × 7 columns

The imported dataset contains:
1338 rows x 7 columns

Draw Scatterplot

```
In [3]: sns.scatterplot(x=ds['bmi'], y=ds['charges'])

Out[3]: <Axes: xlabel='bmi', ylabel='charges'>
```

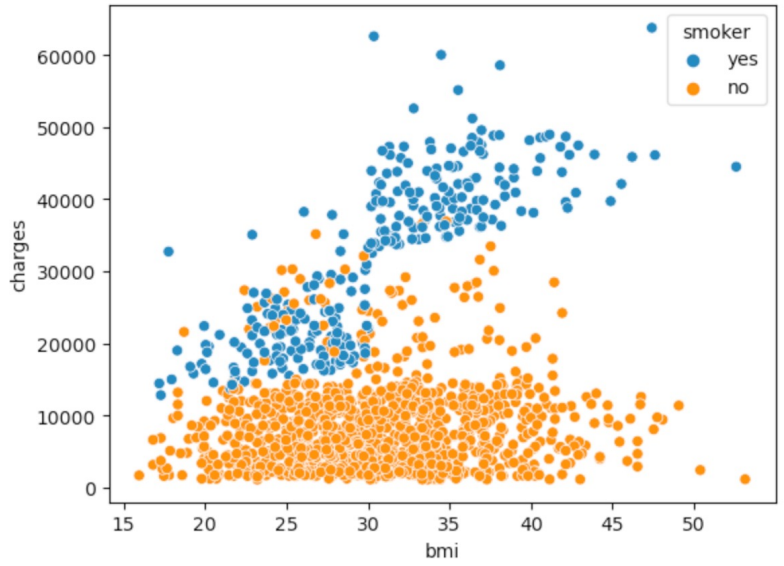


Scatter Plot between Charges and bmi

Scatterplot with hue color

```
In [4]: sns.scatterplot(x=ds['bmi'], y=ds['charges'], hue=ds['smoker'])

Out[4]: <Axes: xlabel='bmi', ylabel='charges'>
```



Scatter Plot between Charges and bmi
Showing the smokers in blue and non smoker in orange

Handling Noisy Data

- **Binning:** known as discretization, is a technique used to handle noisy or continuous data by grouping values into discrete intervals or bins. This can help in reducing the impact of noise and making the data more manageable. Binning can be especially useful when dealing with data preprocessing for machine learning or data analysis.

Discretization

```
In [5]: des = pd.qcut(ds['age'], q= 4).head()
        des

Out[5]: 0    (17.999, 27.0]
        1    (17.999, 27.0]
        2      (27.0, 39.0]
        3      (27.0, 39.0]
        4      (27.0, 39.0]
        Name: age, dtype: category
        Categories (4, interval[float64, right]): [(17.999, 27.0] < (27.0, 39.
        0] < (39.0, 51.0] < (51.0, 64.0]]
```

Binning into 4 equal sized bins

qcut is used to determine how to divide the dataset into 4 groups

Discretization cont.

```
In [6]: ds['age'] = pd.qcut(ds['age'], q=4, labels=['Low', 'MEDIUM', 'HIGH', 'VERY HIGH'])
        ds['age'].head()

Out[6]: 0       Low
        1       Low
        2    MEDIUM
        3    MEDIUM
        4    MEDIUM
        Name: age, dtype: category
        Categories (4, object): ['Low' < 'MEDIUM' < 'HIGH' < 'VERY HIGH']
```

From qcut, we can know the classification of whether the number of cases is low medium high very high

Discretization cont.

```
In [8]: x = ds.keys()
        x

Out[8]: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'], dtype='object')

In [9]: print(ds['age'])

        0             Low
        1             Low
        2          MEDIUM
        3          MEDIUM
        4          MEDIUM
                   ...
        1333         HIGH
        1334          Low
        1335          Low
        1336          Low
        1337    VERY HIGH
        Name: age, Length: 1338, dtype: category
        Categories (4, object): ['Low' < 'MEDIUM' < 'HIGH' < 'VERY HIGH']
```

Dimension Reduction

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of input variables or features in a dataset. The goal is to simplify the dataset while retaining its essential information and patterns. High-dimensional data, where each data point has a large number of features, can be challenging to analyze and may suffer from the curse of dimensionality.

```
In [10]: from sklearn.decomposition import PCA
         import numpy as np
         pca = PCA(n_components=2)
         numerical_cols = ds.select_dtypes(include=[np.number])
         numerical_cols
         pca_result = pca.fit_transform(numerical_cols)
```

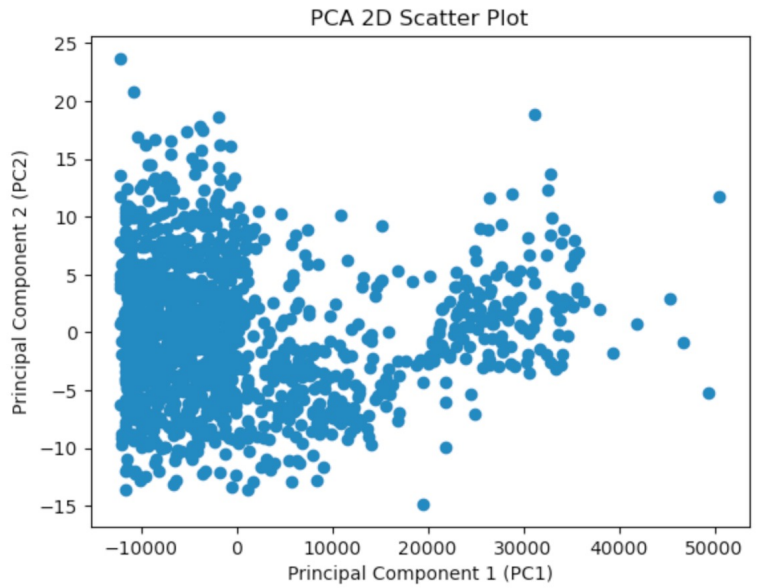
Dimension Reduction Cont.

```
In [10]: from sklearn.decomposition import PCA
         import numpy as np
         pca = PCA(n_components=2)
         numerical_cols = ds.select_dtypes(include=[np.number])
         numerical_cols
         pca_result = pca.fit_transform(numerical_cols)
```

- Here, creating a PCA object with the argument n_components set to 2. To reduce the dimensionality of the data to two principal components.
- numerical_cols = ds.select_dtypes(include=[np.number]): selecting the numerical columns from the dataset ds using the select_dtypes method.
- pca_result = pca.fit_transform(numerical_cols): fits the PCA model to the numerical_cols dataset and transforms it into a lower-dimensional representation. After this line of code, pca_result will contain the transformed data with only two principal components.

Dimension Reduction Cont.

```
In [11]: plt.scatter(pca_result[:, 0], pca_result[:, 1])
         plt.xlabel('Principal Component 1 (PC1)')
         plt.ylabel('Principal Component 2 (PC2)')
         plt.title('PCA 2D Scatter Plot')
         plt.show()
```



create a scatter plot of the data after applying PCA.

References

- Data Source:
- https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset

- Software Tools:
  Anaconda, Jupyter Notebook
- Text Book:
- Visualization Analysis and Design by Tamara Munzner ISBN-10: 9781466508910 • ISBN-13: 978-1466508910 ©2014
- u Python for Data Analysis 2/E by Wes McKinney ISBN-10: 1491957662 • ISBN-13: 1491957660 ©2018
- **Data Preprocessing in Data Mining (Intelligent Systems Reference Library, 72) 2015th**