Big Data & Predictive Analytics
Lab 4 - Exploratory Data Analysis - Visualizing Data with Python

Objectives

In this Lab, you will learn useful techniques that are important in data analytics. At the end, of this practical, you should be able to:

- perform data munging or cleansing for a given dataset
- perform descriptive and exploratory analysis of a dataset
- interpret results from your analyses in order to formulate hypotheses

We consider the **Automobile dataset** again and we aim to carry out an exploratory analysis in order to discover relationships between variables.

Loading Datasets

1. In fact, we have already done most of the tasks in this code in this week's tutorial. To refresh, let us perform the following tasks:

   (a) Save the Automotive dataset into your **local** folder for this module; then use the given Python script `auto_price_clean.py` in which you can find the function `clean_auto` to clean the data. You may want to have a closer look at the way we have removed the outliers in the function `id_outlier`.

   (b) Get the names and data types of the columns in this dataset and examine them to distinguish which variables are numerical or categorical

   Note: Character data is typically treated as categorical where each unique string value is a categorical value of the column. When exploring data sets, it is important to understand the relationships between each category and the label values.

   (c) Modify the function clean_auto so that you remove only remove rows if all column values are missing and that missing numerical values are replaced by their respective column averages and missing categorical values are replaced using the forward fill method. Also remove duplicates for the columns `['price', 'curb-weight', 'width', 'wheel-base']`

   (d) Use your modified function `clean_auto` to get the summary statistics for the clean data with and without removing the outliers.

Create a Pair-Wise Scatter Plot

In this exercise, you will apply a visualization technique known as a scatter plot matrix to the Automotive dataset. Scatter plot matrix methods quickly produce a single overall view of the relationships in a dataset. The scatter plot matrix allows you to examine the relationships between many variables in one view. The alternative of examining a large number of scatter plot combinations one at a time is both tedious and in all likelihood difficult. This is especially the case, as you must remember relationships you have already viewed to understand the overall structure of the data.

2. Consider the following code:

```
## Numeric columns
plot_cols = ["wheel-base",
"width",
"height",
"curb-weight",
"engine-size",
"bore",
"compression-ratio",
"city-mpg",
"price",
"lnprice"]

## Create pair-wise scatter plots
def auto_pairs(plot_cols, df = auto_price):
    import matplotlib.pyplot as plt
    from pandas.plotting import scatter_matrix
    fig = plt.figure(1, figsize=(12, 12))
    fig.clf()
    ax = fig.gca()
    scatter_matrix(df[plot_cols], alpha=0.3, diagonal='kde', ax = ax)
    plt.show()
    return('Done')
```

(a) Run the command auto pairs(plot cols, auto price) and observe the output graph.

(b) Examine the result and note that this plot is comprised of a number of scatter plots. For each variable there is both a row and a column. The variable is plotted on the vertical axis in the row, and on the horizontal axis in the column. In this way, every combination of cross plots for all variables is displayed in both possible orientations. Examine scatter plot matrix, which shows plots of each numeric column verses every other numeric column, and note the following:

- Not surprisingly the features, price and lnprice (log of price) are closely related.
- Many features show significant collinearity, such as wheel-base, width and curb-weight, or curb-weight and engine-size, and city-mpg.

- A number of features show a strong relationship with the label price, such as city-mpg, engine-size, and curb-weight.
- The feature, compression-ratio, has two distinct groupings of values, apparently corresponding to diesel (high compression ratio) and gasoline (low compression ratio) engines. These tightly grouped sets of values act very much like a categorical variable.

Note: The number of scatter plots and the memory required to compute and display them can be a bit daunting. You may wish to make a scatter plot matrix with fewer columns. For example, you can eliminate columns, which are collinear with other columns such as highway.mpg, using only city mpg.

Create Conditioned Histograms

In this exercise, you will create histograms of certain numeric columns in the Automobile dataset. Histograms are a basic, yet powerful, tool for examining the distribution properties of a dataset.

Note: You have already worked with histograms in a previous lab, so here you will focus on **conditioned histograms** of certain columns. A *conditioned histogram* is a histogram of a subset of data conditioned on another variable in the dataset. Often the histogram of a numeric variable is conditioned on a categorical variable.

3. Consider the following code snippet:

```
## Define columns for making a conditioned histogram
plot_cols2 = ["length",
"curb-weight",
"engine-size",
"city-mpg",
"price"]
## Function to plot conditioned histograms
def cond_hists(df, plot_cols, grid_col):
    import matplotlib.pyplot as plt
    import seaborn as sns
    ## Loop over the list of columns
    for col in plot_cols:
        g = sns.FacetGrid(df, col=grid_col, margin_titles=True)
        g.map(plt.hist, col)
        plt.show()
```

(a) Run the function cond hists (auto price, plot cols2, 'drive-
wheels') (b) Examine this series of conditioned plots, and note
the following:

- There is a consistent difference in the distributions of the numeric features conditioned on the categories of drive-wheels.

- The distribution of the values generally increases for length and curb-weight for real wheel drive (rwd) cars, with the values for 4 wheel drive (4wd) and real wheel drive (rwd) overlapping.
- Cars with fwd have the highest city-mpg, with 4wd and rwd in a similar range.
- Generally, 4wd cars have the lowest price, with rwd cars having the widest range.

Note: In this exercise, you have created histograms conditioned on the type of drive wheels. In each case, the conditioning has highlighted different aspects of the relations in these data. Exploring the distributions in the dataset conditioned on other features will likely highlight other aspects of the structure of these data. You may wish to try this on your own.

Create Conditioned Box Plots

In this exercise, you will create conditioned box plots. Box plots are another tool for comparing distributions of conditioned numeric variables. Box plots allow comparison of summary statistics, median and quartiles, as well as to visualize outliers.

4. Consider the following Python code

```
## Create Boxplots of data
def auto_boxplot(df, plot_cols, by):
    import matplotlib.pyplot as plt
    for col in plot_cols:
        fig = plt.figure(figsize=(9, 6))
        ax = fig.gca()
        df.boxplot(column = col, by = by, ax = ax)
        ax.set_title('Box plots of {} bg {}'.format(col, by))
        ax.set_ylabel(col)
        plt.show()
```

The auto boxplot function computes box plots, conditioned on the drive-wheels column, given the name of a numeric column from the auto price dataset.

(a) Run the function auto boxplot(auto price, plot cols2, 'drive-wheels') with the given inputs.

(b) Examine the results and note the following:

Note: The conclusions you can draw from these plots are much the same as from the conditioned histograms. One detail now stands out; the curb-weight of 4wd cars is mostly between that of fwd and rwd cars. As is often the case, a different view or visualization for the same data results in finding new relationships.

Create Conditioned Scatter Plots

Scatter plots are widely used to examine the relationship between two variables. In this exercise, you will explore methods to show the relationship between more than two variables on a two dimensional scatter plot. First, you will apply colour as a method to examine multiple dimensions. By adding colour to a scatter plot, you can effectively project three dimensions onto a two-dimensional plot. Next, you will create and examine conditioned scatter plots. By conditioning multiple dimensions, you can project several additional dimensions onto the two-dimensional plot.

You will not use point shape as a differentiator in this exercise, but keep in mind that shape can be as useful as colour. Additionally, shape may be easier for the significant fraction of the population who are colour blind.

Note: Be careful when combining methods for projecting multiple dimensions. You can easily end up with a plot that is not only hard to interpret, but even harder for you to communicate your observations to your colleagues. For example, if you use three conditioning variables, plus color and shape, you are projecting seven dimensions of your dataset. While this approach might reveal important relationships, it may just create a complex plot.

5. Consider the following Python code:

```
## Define columns for making scatter
plots plot_cols3 = ["length",
             "curb-weight",
             "engine-size",
             "city-mpg"]
## Create scatter plot
def auto_scatter(df, plot_cols):
    import seaborn as sns
    import matplotlib.pyplot as plt
    for col in plot_cols:
        g = sns.FacetGrid(df, margin_titles=True, hue="fuel-type",\
            palette={"diesel":"red", "gas":"blue"}, size=8, aspect=1.5)
        g.map(plt.scatter, col, "price")
        g.add_legend()
        plt.show()
```

(a) Run the function auto scatter(auto price, plot cols3) for the given inputs.

(b) Examine these plots, and note the following:

- Each of the four variables plotted against price shown a strong trend, indicating they might be good predictors of price.
- For the first three plots, diesel cars appear to be on the same trend as gasoline cars. However, a diesel car will tend to be more expensive to achieve the same city-mpg.