Task Document - Week 10

Note to learner*: You MUST complete these exercises and upload your answers to the academy weekly, for your lecturer to review your progress and provide you feedback.*

Exploratory Data Analysis Workshop Exercises

This week, we will be attempting some exploratory data analysis, using both non-graphical and graphical methods. Please work through the following exercises, and then complete the tasks at the end of this document.

Following is the Google Colab Notebook for Week 10.

https://colab.research.google.com/drive/1seW6cAuYaVS6QMnSmI-eZ0enrjHd5LIC?usp=sharing

Exercise 1

First load the diamonds data from CSV file into Python Pandas dataframe.

```
import pandas as pd
url = 'https://raw.githubusercontent.com/tidyverse/ggplot2/main/data-raw/diamonds.csv'
diamonds = pd.read_csv (url)
print(diamonds.shape)                          #(53940, 10)
print(diamonds.head())
```

If you wish,you may view the data to familiarise yourself with the loaded data. Remember that when beginning to explore a new dataset, it is valuable to use the *shape* function to provide details on the dimensionality and structure of the data. The shape of diamonds data is 53940 rows and 10 columns. Use diamonds.head() statement to view the column names. Note that first column shown by the pandas dataframe is for indexing only.

Once you have identified the type of data that make up the “diamonds” dataset, we will begin to examine some of the individual features, through univariate analysis. Using **ggplot2** provides us with a much larger feature set, which allows us to explore the data in moredetail. To begin with, we will investigate how frequently each colour of diamond occurs within the dataset. Use the following command:

```
        from plotnine import *
        ggplot(data = diamonds) + geom_bar(mapping = aes(x = 'color'))
```

Notice that we conjoin elements within a **ggplot2** command using the + symbol. Remember that in this case the feature “color” is an ordinal feature, so even though the data is categorical, we can still derive information from the distribution that is presented (diamond colour ranges from “D”, meaning colourless or clear, to the later letters which represent an increasingly yellow hue). The bar chart is a good way of representing categorical data but is not suitable for continuous data. For that,we must use a histogram. As an example, we shall examine the feature “carat”. Use the following command:

```
ggplot(data = diamonds) + geom_histogram(mapping = aes(x = 'carat'), binwidth = 0.2)
```

Note that the term *binwidth* dictates the width of the bins. A smaller value for *binwidth* produces a more granular representation of the data. Try the command below to see how the plot changes:

```
ggplot(data = diamonds) + geom_histogram(mapping = aes(x = 'carat'), binwidth = 0.1)
```

The bin width should be set at an appropriate size for the purposes of the work that is being performed. If aesthetics is the priority, then trial and error using various widths can be helpful. In producing the histogram, two things should become apparent: the first, that the data has an approximately Poisson distribution, and the second, that there are some outliers. Given these two facts, a visualisation that will help to further understand the data series is a box plot.

Before we do that however, it is important that we remember never to trust the data implicitly during the exploration phases, and to always try to identify the assumptions we are making when dealing with any data. Consider the following exercise. First use the following commands to create two samples from the “carat” data series:

```
import numpy as np

np.random.seed(22)

sample1 = diamonds['carat'].sample(n=1000,replace=False)

sample2 = diamonds['carat'].sample(n=1000,replace=False)
```

Note: the set seed function provides a form of “key” that is used when sampling and sub-setting data, in order to better ensure reproducibility. This will be discussed further in future when creating training and test data from datasets.

After we have the samples, each must be coerced into a data frame format so that **ggplot2** is able to use the data. To achieve this, use the following commands:

```
sample1_df = sample1.to_frame()
```

```
sample2_df = sample2.to_frame()
```

Now, use the following command to create the histogram for the first sample, and then modify the code to create the correct histogram for the second sample as well:

```
ggplot(data = sample1_df) + geom_histogram(mapping = aes(x = 'sample1', binwidth =
0.1))
```

You should notice that while the profile of the samples is similar, there will also be many distinct differences. Consider then that almost all data that you ever handle will be just one sample out of many possible samples, and the dangers of assuming absolute truth from any data set become clear.

Moving back to the box plot, to create this visualisation for the "carat" feature, use the following command:

```
ggplot(data = diamonds) + geom_boxplot(mapping = aes(x='color',y = 'carat'))
```

Notice that the outliers that we identified are clearly shown beyond the boundaries of the "whisker". To make them stand out even more, use the following command:

```
ggplot(data = diamonds) + geom_boxplot(mapping = aes(x='color',y =
'carat'),outlier_colour="red")
```

The presence of those outliers is intriguing. Let us try and establish whether these very high-carat diamonds have an equivalently high price, by creating a scatter plot mapping the two features. Use the following command:

```
ggplot(data = diamonds) + geom_point(mapping = aes(x = 'carat', y = 'price'))
```

We can see that there is a huge amount of variance among diamonds of the same or similar carat. This suggests that there may be one or more additional features responsible for this variance. Using **ggplot2** allows us to add further dimensions into a two-dimensional plot through the use of pre-attentive attributes, such as shape, colour and size. For this task, we shall use different colours to map the cut quality of the diamond onto the scatter plot, using the following command:

```
ggplot(data = diamonds) + geom_point(mapping = aes(x = 'carat', y = 'price', color = 'cut'))
```

Now we can see that in fact price appears to be a function of carat and the quality of cut. The discovery of this relationship is why multivariate EDA is so powerful. If we were to build a model to try and predict the price of diamonds, we now know that "carat" and "cut" are two of the features that should be selected for inclusion.

When performing multivariate EDA, we will sometimes have to analyse the relationship between two categorical features, which prohibits the use of scatter plots. One method that we can use in lieu are frequency plots, which present a grid representing the possible combinations of two categorical features, with the size of objects in the plot showing the frequency occurrence of each. To identify the spread of observations between category combinations of cut and clarity, use the following command:

```
ggplot(data = diamonds) + geom_count(mapping = aes(x = 'cut', y= 'clarity'))
```

Notice that the larger “nodes” signify the most frequent combinations.

Exercise 2

One important part of EDA is the identification of correlations between the independent and dependent features, and covariance between independent features in the dataset. A method that can be used to achieve this is the production of correlation matrix, in a either a non-graphical or graphical format. An issue that we face, however, is the presence of categorical features within the data, which will inhibit the production of a correlation matrix. Fortunately, the features in question (“cut”, “color” and “clarity”) are ordinal, meaning that they have a natural order, and can therefore be coded as natural numbers. By default, they should be stored as *factors*, but we will convert themto integers.

Whenever this form of transformation is applied, it is essential that the data is investigated thoroughly, and appropriate domain knowledge utilised, to ensure that there are no unintended consequences. For example, when performing correlation analysis, we need to the directionality of all the feature to be the same (i.e. “carat” becomes a ‘better’ value as it rises, as does price). Using the *diamonds.describe()* function to examine the data, it can be seen that the level order for the factors of “cut” and “clarity” possess this appropriate directionality, where the lower levels correspond with a less desirable category. However, this directionality is reversed for the feature “color”, where the lowerlevels correspond to a more desirable trait (i.e. less colour contamination).

Once that has been taken care of, we can convert the factors to natural numbers. For the feature “color”, we can perform this transformation using the following command:

```
diamonds['color'] = diamonds['color'].astype('category')
diamonds['color'] = diamonds['color'].cat.codes
```

When viewing the data, you will see that "color" is now represented numerically. Now, adapt the command above to apply the same transformation for "clarity" and "cut".

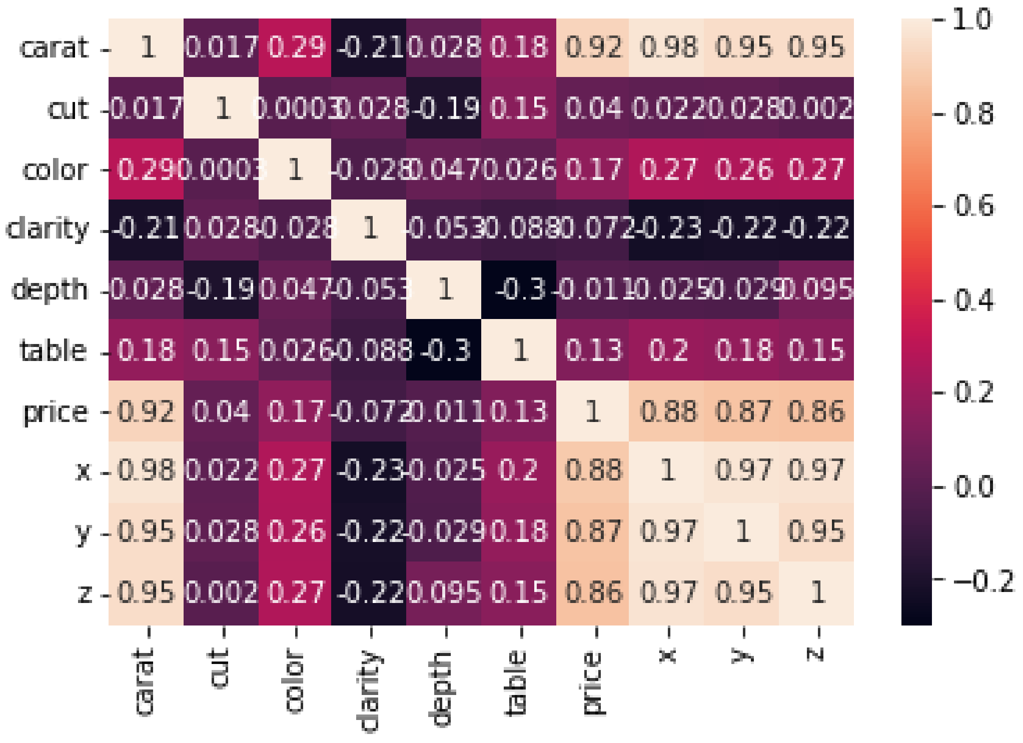
To create a correlation matrix, showing the Pearson correlation coefficients between the features, we can use the *corr* function of the Pandas dataframe. Use the following command:

```
corr = diamonds.corr(method = "pearson")
corr
```

Notice that the "method" parameter allows for the measure of correlation to be set. While the correlation matrix that is printed to the console illustrates those raw figures, it does not present the best platform for the identification of patterns and trends. For that, a more graphical approach should be employed. A useful package that provides this functionality is **statistical data visualization (seaborn)** and **matplotlib**. Usethe following commands to download and install the package, and produce a visualisation that can be used for effective graphical EDA:

```
import seaborn as sns
import matplotlib.pyplot as plt
heatmap = sns.heatmap(corr, annot=True)
plt.show()
```

Upon completion, you should find yourself with a figure like the one shown below.



The components of the plot can be broken down as follows:

- The plot shows a 10 x 10 square matrix and color-fills each cell based on the correlation coefficient between two variables.
- The values inside the cells represents the correlation coefficients between each pair of features and the significance of each. A strong positive correlation is generally given to be any coefficient value of greater than 0.6 (and likewise, a strong negative coefficient can be considered to be any value lower than - 0.6). The corresponding correlation coefficient for carat and price is indicated by the 'new colonial yellow' box, on the figure above.
- Using this form of graphical EDA is an effective means for discovering potential predictors and identifying covariates. Let us assume that we are building a model which aims to predict diamond price, therefore using that feature as the response variable. As was identified earlier, carat is strongly correlated, but so are the features "x", "y" and "z". These features relate to the physical dimensions of the diamond, in the x, y and z planes. However, we can see that all three are highly correlated covariates of each other and may therefore cause issues with multicollinearity. Because of the high degree of interdependency however, there is very little information lost when two of the features are stripped from the dataset, so that would be the recommended course of action, prior to any model development taking place. Some covariance can be tolerated between independent features, but correlation coefficients in excess of around 0.8 (or -0.8) should certainly warrant further attention.

Notice that we should not discount features just because they are not strongly correlated with the response variable. As we saw earlier in the workshop, while the cut of a diamond is not necessarily a strong predictor of price in isolation, it would appear to be a moderator between carat and price and should therefore be included in any model that is developed.

The final graphical EDA method that we will explore today is the quantile-normal plot. This plot is used to detect how close to approximating a normal distribution the data within a particular feature are, and hence act as a guide to whether certain assumptions have been met, and which techniques are available for use. To produce a quantile-normal for the "depth" feature, use the following commands:

```
!pip install seaborn_qqplot
from seaborn_qqplot import pplot
from scipy.stats import gamma
pplot(diamonds, x="depth", y=gamma, kind='qq', height=4, aspect=2,
display_kws={"identity":True})
```

Notice that the line created using the 'identity:True' parameter signifies a normal distribution, and any deviation from this line shows non-normality. Adapt the above commands for the "carat" feature.What do you notice? Which features follows the normal distribution most closely?

Exercise 3

For the tasks below, happiness.csv is provided, and load it as Pandas dataframe. Remove the feature containing the country names.

1. Perform some univariate EDA on the features within the dataset. Complete the table below(the first row has been pre-completed):

Feature	Skewness (positive or negative)	Kurtosis (positive or negative)	Appropriate measure of central tendency (MoCT)	MoCT value	Outliers?
GDP per capita	Negative	Negative	Median	0.96	No
Generosity					
Healthy life expectancy					
Perceptions of corruption					

2. Produce a correlation chart for the dataset. When developing a model to predict the overall satisfaction score:
   2.1. Which features would act as the strongest predictor?
   2.2. Which features may you wish to remove from the dataset, and why?

3. Explain how the techniques covered in this week, and in previous weeks, can help to satisfy:
   3.1. The assumption of feature independence.
   3.2. The assumption of observational independence.
   3.3. The assumption of the approximation of normality.
   3.4. The assumption of accurate data.