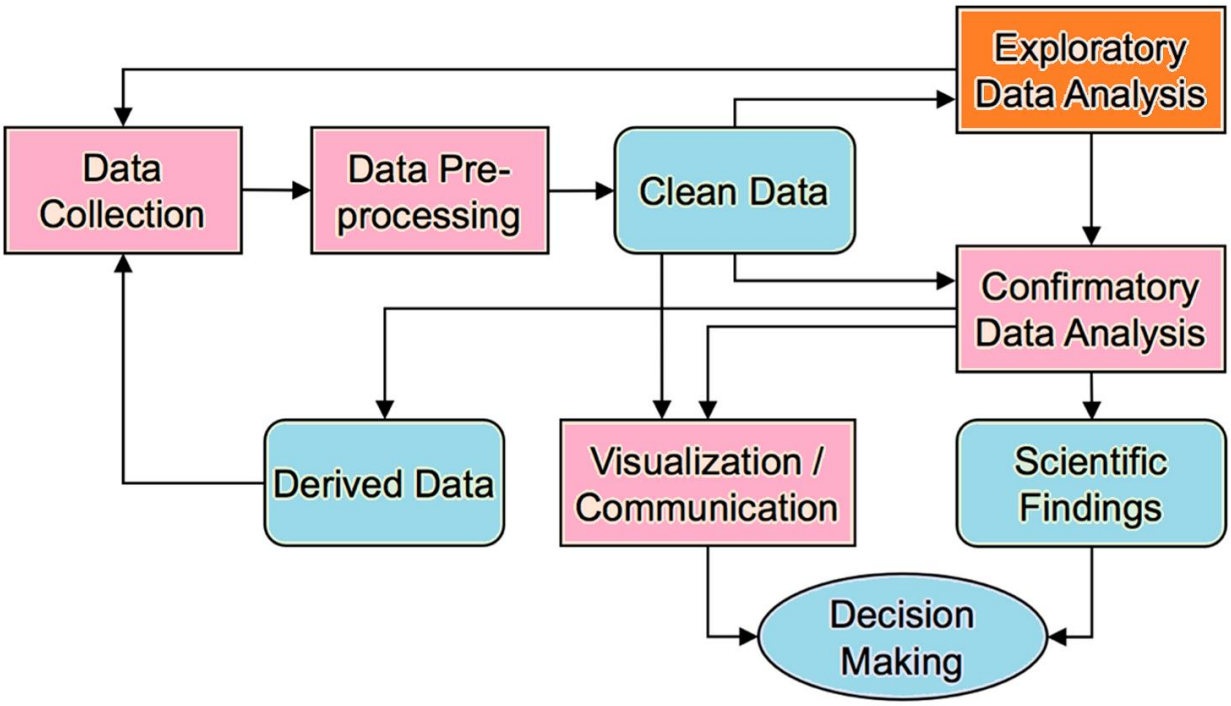
Chapter 1: Prepare the problem, Structural Investigation, Quantitative Investigation, Outliers, Content Investigation

Introduction

In this chapter, we will cover the topics related to Exploratory Data Analysis. But before that, we will be working on a problem statement. A problem will be addressed in terms of a use case scenario from a dataset. Then that dataset will be analyzed thoroughly.

Initial analysis will be done on the datasets to investigate some of the data features. EDA is a data analysis method that allows data scientists to explore and summarize the important characteristics of a dataset by visualizing the findings. EDA describes how to best manipulate the data sources. When used with big data, it allows data scientists to discover patterns, spot anomalies, and test assumptions. It can be viewed as:



In the upcoming exercises, we will be analyzing the dataset having price records of houses in Miami. Different features associated with the trend of prices along with the nature of data structure, content, and quality will be observed. Moreover, outliers will also be covered in the upcoming exercises.

You can learn more about the EDA from here: https://youtu.be/YRBdTw9TZPE

Investigations

- Structure Investigation.
- Quality Investigation.
- Content Investigation.

Preparing the problem

If you have a large dataset with more than 100 features, it will be overwhelming to try to generate visualizations for all of them. In such a case, variable selection will become very important. The EDA is often seen as the hardest, and most time-consuming part of the process. But it is highly recommended for anyone who's working on large data with more than 100 features to do variable selection first before jumping into EDA.

In this chapter, we will be exploring the dataset from Miami. The dataset contains information on 13,932 single-family homes sold in Miami in 2016. Besides publicly available information, the

dataset creator Steven C. Bourassa has added distance variables, aviation noise as well as latitude and longitude. This dataset will be explored in terms of its features. Numeric and categorical variables will be extracted from it.

Exercise 1.01: Importing the Libraries and Loading the Data

In this exercise, we will import the essential libraries and load the Miami housing dataset. Initial data cleaning will be done here:

Perform the following steps to implement this exercise:

1. Open collab Notebook and import the required libraries as shown below:

Import the visualization libraries alongwith the other required ones.

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

%matplotlib inline
```

```
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

import certifi
from sklearn.datasets import fetch_openml
```

2. Then load the dataset using following commands:

```
# Load the dataset
df =
pd.read_csv('https://raw.githubusercontent.com/fenago/datawrangling/main/miam
i-housing.csv')
df.sample(5)
df.head()
```

It will give the following output:

	LATITUDE	LONGITUDE	PARCELNO	SALE_PRC	LND_SQFOOT	TOT_LVG_AREA	SPEC_FEAT_VAL	RAIL_DIST	OCEAN_DIST	WATER_DIST	CNTR_DIST	SUBCNTR_DI	HWY_DIST	age	avno60plus	month_sold	struct
0	25.891031	-80.160561	622280070620	440000.0	9375	1753	0	2815.9	12811.4	347.6	42815.3	37742.2	15954.9	67	0	8	
1	25.891324	-80.153968	622280100460	349000.0	9375	1715	0	4359.1	10648.4	337.8	43504.9	37340.5	18125.0	63	0	9	
2	25.891334	-80.153740	622280100470	800000.0	9375	2276	49206	4412.9	10574.1	297.1	43530.4	37328.7	18200.5	61	0	2	
3	25.891765	-80.152657	622280100530	988000.0	12450	2058	10033	4585.0	10156.5	0.0	43797.5	37423.2	18514.4	63	0	9	
4	25.891825	-80.154639	622280100200	755000.0	12800	1684	16681	4063.4	10836.8	326.6	43599.7	37550.8	17903.4	42	0	7	

Structural Investigation

This step allows user to explore the general structure of a dataset and understand what types of features it contains.

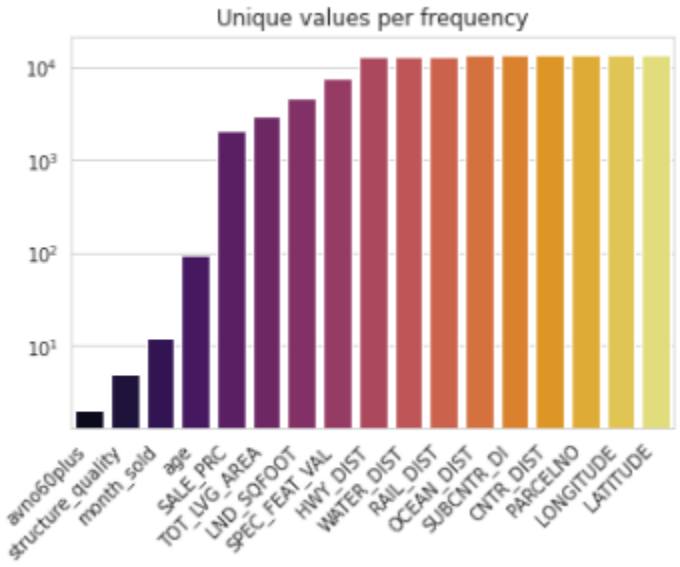
The structure of any dataset is important to understand the possible choices of feature extraction. The dimensions of columns and rows affects various features or attributes that can be used for classification or prediction, as well as other statistical analyses. We can also draw conclusions based on which columns are excluded from our data, e.g., if they are not represented at all in our dataset then we don't need to consider them when using those columns for classification or prediction. So, it is important to have a look at the general structure of the data.

This investigation covers these two parameters:

1. Structure of Non-numerical Features
2. Structure of Numerical Features

We will be covering structural investigation in the upcoming exercise.

The following output is obtained from the barplot, numerical features analysis having unique values per frequency:



The general structure thus obtained is as:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13932 entries, 0 to 13931
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   LATITUDE           13932 non-null  float64
 1   LONGITUDE          13932 non-null  float64
 2   PARCELNO           13932 non-null  int64  
 3   SALE_PRC           13932 non-null  float64
 4   LND_SQFOOT         13932 non-null  int64  
 5   TOT_LVG_AREA       13932 non-null  int64  
 6   SPEC_FEAT_VAL      13932 non-null  int64  
 7   RAIL_DIST          13932 non-null  float64
 8   OCEAN_DIST         13932 non-null  float64
 9   WATER_DIST         13932 non-null  float64
 10  CNTR_DIST          13932 non-null  float64
 11  SUBCNTR_DI         13932 non-null  float64
 12  HWY_DIST           13932 non-null  float64
 13  age                13932 non-null  int64  
 14  avno60plus         13932 non-null  int64  
 15  month_sold         13932 non-null  int64  
 16  structure_quality  13932 non-null  int64  
dtypes: float64(9), int64(8)
memory usage: 1.8 MB
```

Exercise 1.02: Structure Investigation

You are required to analyze the given dataset in terms of its general Structure-based features. Load the dataset and import the required libraries. Look for the data cleaning process. Then move forward to perform the Structure-based analysis of the data.

Perform the following steps to implement this exercise:

1. Open collab Notebook and import the required libraries as shown below:

Import the visualization libraries alongwith the other required ones.

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

%matplotlib inline
```

```
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

import certifi
from sklearn.datasets import fetch_openml
```

2. Then load the dataset using following commands:

```
# Load the dataset
df =
pd.read_csv('https://raw.githubusercontent.com/fenago/datawrangling/main/miami-housing.csv')
df.sample(5)
df.head()
```

3. Then analyze the size of dataset and count the number of datatypes present in the data as shown below:

```
# Show size of the dataset
df.shape
```

```
# Count how many times each data type is present in the dataset
pd.value_counts(df.dtypes)
```

It will produce the following output:

```
float64    9
int64      8
dtype: int64
```

4. Then compute the number of unique entries while identifying numerical features as shown below:

```
In [ ]:
# For each numerical feature compute number of unique entries
unique_values = df.select_dtypes(include='number').nunique().sort_values()
plt.figure(figsize=(15, 4))
sns.set_style('whitegrid')
```

Following output will be produced:

```
<Figure size 1080x288 with 0 Axes>
```

```
g = sns.barplot(x=unique_values.index, y=unique_values, palette='inferno')
g.set_yscale("log")
g.set_xticklabels(g.get_xticklabels(), rotation=45, horizontalalignment='right')
g.set_title('Unique values per frequency')
plt.show()
```

5. The type of features will be explored here:

```
df.info()
df.describe()
```

Following outputs will be produced:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13932 entries, 0 to 13931
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   LATITUDE           13932 non-null  float64
 1   LONGITUDE          13932 non-null  float64
 2   PARCELNO           13932 non-null  int64  
 3   SALE_PRC           13932 non-null  float64
 4   LND_SQFOOT         13932 non-null  int64  
 5   TOT_LVG_AREA       13932 non-null  int64  
 6   SPEC_FEAT_VAL      13932 non-null  int64  
 7   RAIL_DIST          13932 non-null  float64
 8   OCEAN_DIST         13932 non-null  float64
 9   WATER_DIST         13932 non-null  float64
 10  CNTR_DIST          13932 non-null  float64
 11  SUBCNTR_DI         13932 non-null  float64
 12  HWY_DIST           13932 non-null  float64
 13  age                13932 non-null  int64  
 14  avno60plus         13932 non-null  int64  
 15  month_sold         13932 non-null  int64  
 16  structure_quality  13932 non-null  int64  
dtypes: float64(9), int64(8)
memory usage: 1.8 MB
```

Quality Investigation

The first step in any data analysis process is to perform the checks of the quality on the data beforehand. In this case, we will be focusing on the overall quality of dataset. Before looking

into the details, we can look at some general statistics of this dataset and check if there are any unexpected entries or duplicate rows.

Removal of Duplicates

Duplicates are entries that represent the same sample point multiple times. It is common when two people take a measurement at the same location of your dataset, or even when two recordings are made by different people under similar conditions at the same time. Ways exist to eliminate duplicates from your records: delete all duplicates in a dataset and/or detect duplicates only as necessary.

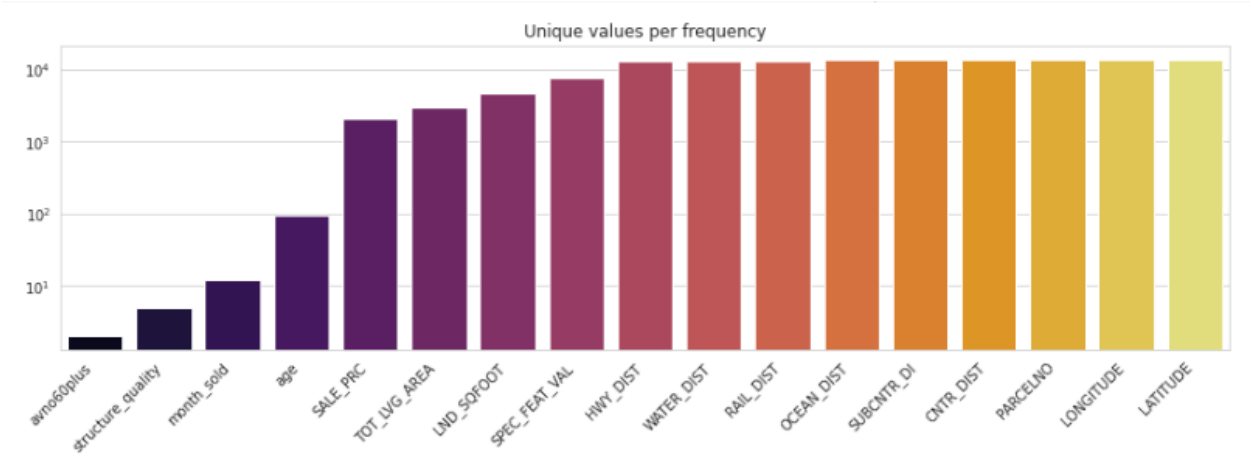
Duplicates can be dropped through .***drop_duplicates().***

Missing Values

Missing data can be a critical problem with any dataset, but it is very important to make sure they are handled correctly when cleaning or analyzing the data. When you begin data cleaning, missing values are a big problem. For some datasets, tackling first the features and then the samples might be better. Furthermore, the threshold at which it is decided to drop missing values per feature or sample should change in different circumstances.

Per Sample

There are multiple options to consider missing values per sample. The most straightforward one is to simply visualize the result of ***df.isna().***



Another better approach is to use the missingno library to get the plot.

Per Feature

For missing values per feature, some pandas features can be used to quickly identify the ratio of missing values per feature. It can be implemented as:

```
df.isna().mean().sort_values().plot(
```

```
    kind="bar", figsize=(15, 4),
    title="Percentage of missing values per feature",
    ylabel="Ratio of missing values per feature");
```

Unwanted Entries and recording Errors

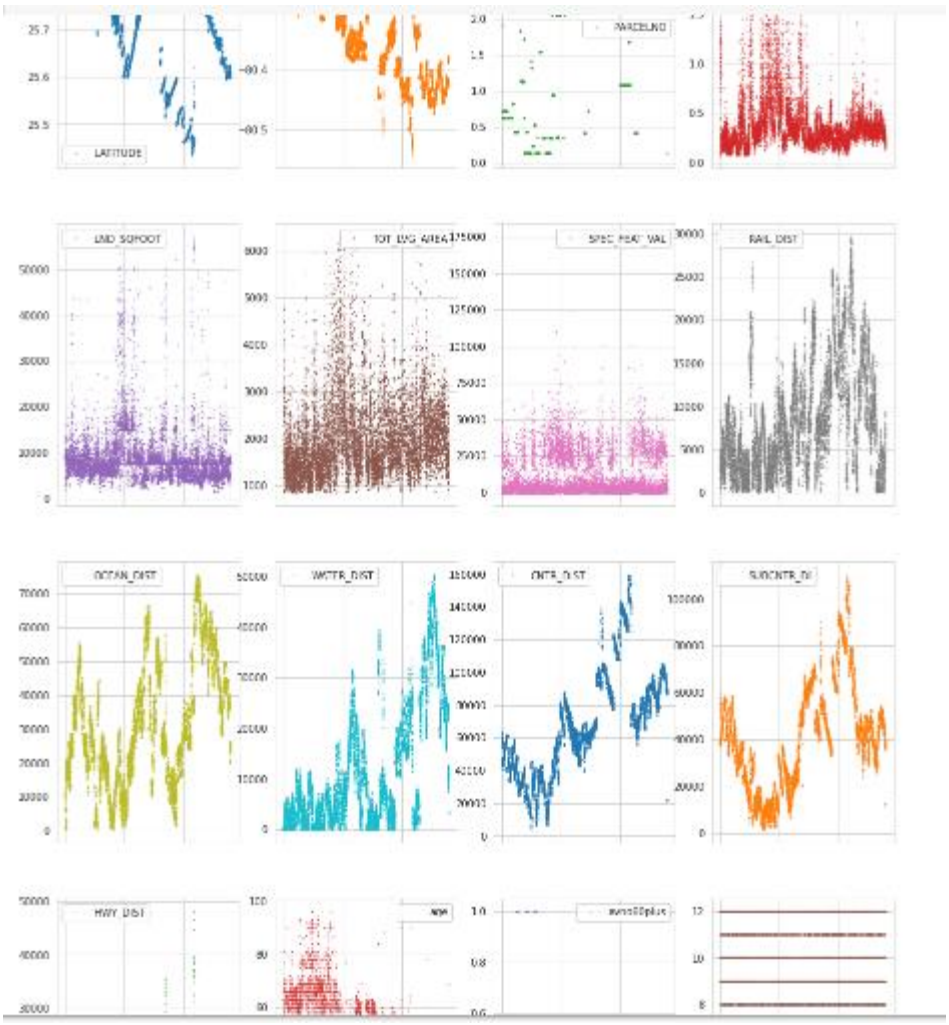
A dataset without unwanted entries is clean and accurate, but it's hard to tell if a dataset can contain such "**unwanted entries**". Unwanted entries can be caused by incorrect or incomplete recordings or by changes in the data after being defined, for instance, because of adding information to a dataset. They need to be removed.

Numerical Features

You can use the ***.plot***() function to show the results of your data analysis in the form of a scatter plot, histogram, frequency distribution, or bar chart. To get started, pass it the name of one or more parameters to customize your plot. The following parameters can be used:

- **lw=0:** lw stands for line width.
- **marker=".":** Instead of lines, we tell the plot to use . as markers for each data point
- **subplots=True:** Subplots tell pandas to plot each feature in a separate subplot
- **layout=(-1, 4):** This parameter tells pandas how many rows and columns to use for the subplots.
- **figsize=(15, 30), markersize=1:** To make sure that the figure is big enough we recommend having a figure height of roughly the number of features, and adjusting the markersize accordingly.

```
df_X.plot(lw=0, marker=".", subplots=True, layout=(-1, 4),
          figsize=(15, 30), markersize=1);
```



We will be handling the duplicates and missing values in the upcoming exercise.

Non-numerical Features

As our dataset has no non-numerical features, it has displayed the following result:

0
1
2
3
4

Exercise 1.03: Quality Investigation

You are required to analyze the given dataset in terms of its Quality-based features. Load the dataset and import the required libraries. Look for the data cleaning process. Then move forward to perform the Quality-based analysis of the data.

First, perform the steps mentioned in Exercise 1.01 and 1.02 to cover the features of structure analysis, and then perform the following steps for quality investigation on the given data:

1. Firstly, Check number of duplicates while ignoring the index feature as shown below:

```
# Check number of duplicates while ignoring the index feature
n_duplicates = df.drop(labels=['PARCELNO'], axis=1).duplicated().sum()

print(f"You seem to have {n_duplicates} duplicates in your database.")
```

Output will be as follows:

```
You seem to have 0 duplicates in your database.
```

2. Then extract the column names of the features except “PARCELNO” and drop duplicates based on “columns_to_consider” as shown below:

```
In [ ]:
#  Extract column names of all features, except 'PARCELNO' - unique feature
columns_to_consider = df.drop(labels=['PARCELNO'], axis=1).columns

# Drop duplicates based on 'columns_to_consider'
df.drop_duplicates(subset=columns_to_consider, inplace=True)
df.shape
```

Following output will be produced:

```
(13932, 17)
```

3. Then check for the missing values. Follow the code statements as shown below:

```
plt.figure(figsize=(15, 4))
sns.set_style('whitegrid')

g = sns.barplot(x=unique_values.index, y=unique_values, palette='inferno')
g.set_yscale("log")
g.set_xticklabels(g.get_xticklabels(), rotation=45, horizontalalignment='right')
g.set_title('Unique values per frequency')
plt.show()
```

4. For quality investigation, sampling analysis for missing values is done here through the visuals. It is the case of “Per Sample”:

```
plt.figure(figsize=(15, 8))
sns.set_style('whitegrid')

g = sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

g.set_xlabel('Column Number')
g.set_ylabel('Sample Number')
```

```
[12] !pip install missingno
     import missingno as msno
     msno.matrix(df, labels=True, sort='descending', color=(0.27, 0.52, 1.0));
```

```
[13]
     df = df.dropna(thresh=df.shape[1] * 0.80, axis=0).reset_index(drop=True)
     df.shape
```

5. For quality investigation, analysis based on features is done here through the visuals As a next step, let's now look at the number of missing values per feature.

```
[ ]: df.isna().mean().sort_values().plot(
         kind="bar", figsize=(15, 4),
         title="Percentage of missing values per feature",
         ylabel="Ratio of missing values per feature");
```

```
[ ]: df = df.dropna(thresh=df_X.shape[0] * 0.85, axis=1)
     df.shape
```

Following output will be produced:

```
(13932, 17)
```

6. Now use pandas’s plot feature to plot the global view of the dataset:

```
[ ]: df.plot(lw=0, marker=".", subplots=True, layout=(-1, 4),
             figsize=(15, 30), markersize=1);
```

7. Finally have some insight into the non-numerical features of dataset as shown below:

```
In [ ]:
# Display non-numerical features
df.select_dtypes(exclude="number").head()
```

Content Investigation

The general structure and quality of the data have been observed in the last section. Let's now explore more and take a look at the actual content. Such an investigation is mostly done feature by feature. But it can become cumbersome in the case of 20-30 features.

The following three approaches will give you an idea about the content-related investigations:

Feature Distribution

One of the methods to analyze the content of the dataset involves the value distribution of each feature. It helps to guide EDA as well and provides a lot of information that can help in data cleaning as well as feature transformation. And it can be implemented for the numerical features through the histogram plots. Pandas offer a built-in function for histograms as depicted here:

df.hist(bins=25, figsize=(15, 25), layout=(-1, 5), edgecolor="black")

Most frequent entry

Dataset may contain some features that contain entries of just one kind. Using the .mode() function, the ratio of the most frequent entry for each feature can be extracted and we can visualize that information.

Skewed value distributions

Some numerical features can show non-Gaussian distributions. In such a case, a data scientist may think to transform these values to make them more normally distributed. A "log transformation" can be done in the case of right-skewed data.

Feature Patterns

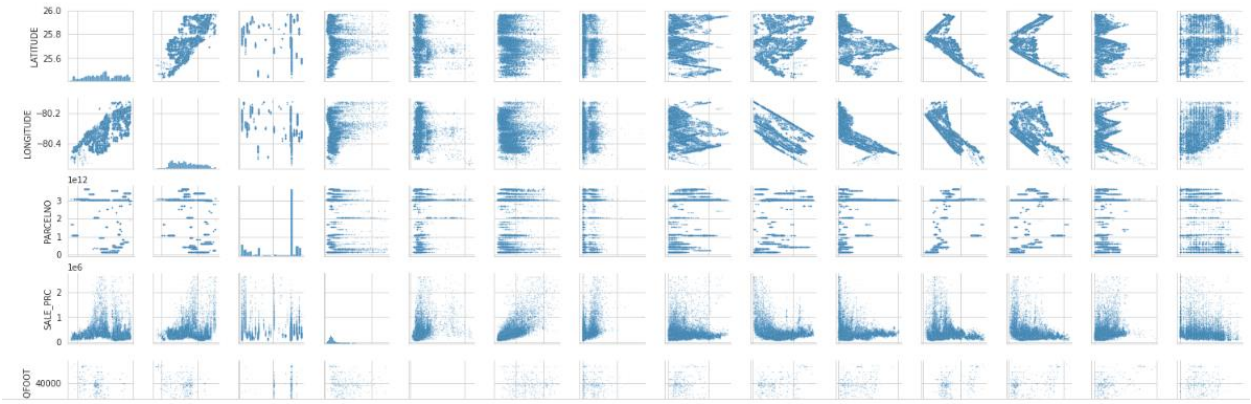
The investigation of feature-specific patterns is the next step. The goal includes the following two cases:

1. Data Inspection is the analysis of the quality, and completeness of the information that is stored. It must be achieved first.
2. The relationships between features will help us in understanding the dataset.

Continuous Features:

Seaborn's "pairplot" can be used to visualize the relationships between the continuous features. It can be a time-taking process to create all subplots. So, it is recommended that it

may not be used with more than 10 features. In our case, we have 17 features, we can somehow use pairplot. This can be seen in our upcoming exercise.



Discrete and Ordinal features

It is more difficult to find patterns in the discrete or ordinal feature. But some quick pandas and seaborn features can help to get a general overview of a dataset.

```
# Create a new data frame that doesn't contain the numerical continuous features

df_discrete = df[cols_continuous[~cols_continuous].index]

df_discrete.shape
```

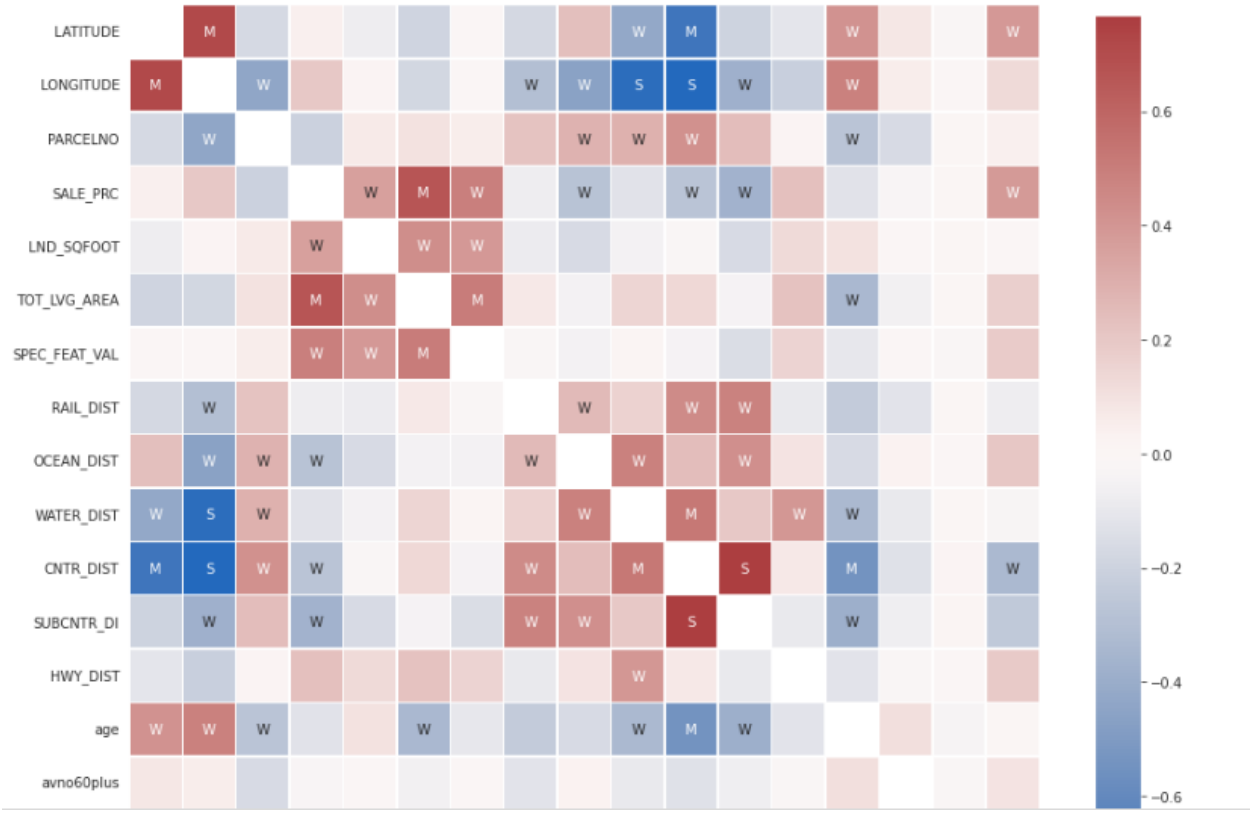
Feature Relationships

It involves the relationships between the features. It can be done through the ".**corr**()" method. It can be implemented as follows:

```
# Computes feature correlation

df_corr = df.corr(method="pearson")
```

The "**Spearman**" method can also be used instead of the "**Pearson**" method, depending upon the dataset. Whereas the Pearson correlation evaluates the linear relationship between two continuous variables. The Spearman correlation evaluates the monotonic relationship based on the ranked values for each feature. And to help with the interpretation of this correlation matrix, let's use **seaborn's** .**heatmap**() to visualize it.



Exercise 1.04: Content Investigation

You are required to analyze the given dataset in terms of its Content-based features. Load the dataset and import the required libraries. Look for the data cleaning process. Then move forward to perform the Content analysis of the data.

First, perform the steps mentioned in Exercise 1.01 and 1.02, and 1.03 to cover the features of structure analysis, and Quality Investigation and then perform the following steps for Content investigation on the given data:

1. Firstly, plot the histogram for each numerical feature in a separate subplot:

```
In [ ]: df.hist(bins=25, figsize=(15, 25), layout=(-1, 5), edgecolor="black")
        plt.tight_layout();
```

2. Then investigate the patterns. Create mask to determine the numerical features having almost 25 unique features. Then create another data frame that will contain the continuous features only as shown below:

In [ ]:

```
cols_continuous = df.select_dtypes(include="number").nunique() >= 25

# Create a new dataframe which only contains the continuous features
df_continuous = df[cols_continuous[cols_continuous].index]
df_continuous.shape

sns.pairplot(df_continuous, height=1.5, plot_kws={"s": 2, "alpha": 0.2});
```

3. Then create a new data frame that does not contain the numerical continuous features, as shown below:

```
[23]
     df_discrete = df[cols_continuous[~cols_continuous].index]
     df_discrete.shape
```

Following output will be produced:

```
(13932, 3)
```

4. Then perform the computation of feature correlation. Create labels for the correlation matrix and then plot it, as shown below:

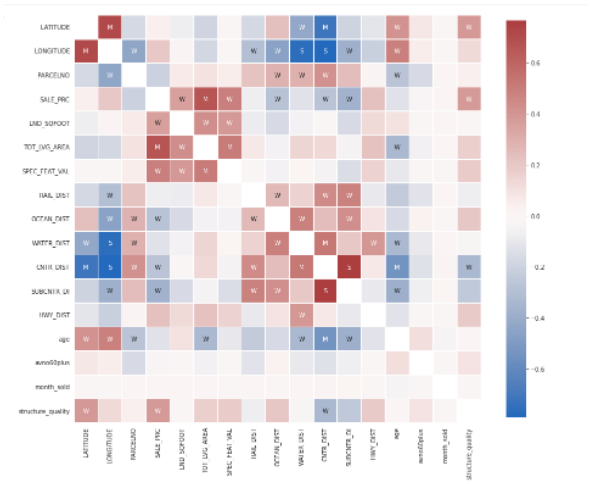
In [ ]:

```
# Computes feature correlation
df_corr = df.corr(method="pearson")

# Create labels for the correlation matrix
labels = np.where(np.abs(df_corr)>0.75, "S",
                  np.where(np.abs(df_corr)>0.5, "M",
                           np.where(np.abs(df_corr)>0.25, "W", "")))

# Plot correlation matrix
plt.figure(figsize=(15, 15))
sns.heatmap(df_corr, mask=np.eye(len(df_corr)), square=True,
            center=0, annot=labels, fmt='', linewidths=.5,
            cmap="vlag", cbar_kws={"shrink": 0.8});
```

Following output will be produced:



5. Then create a mask to remove the diagonal and the upper triangle. Meanwhile, stack all correlations. Then show the lowest and highest correlations in the matrix as shown below:

```
[ ]:
# Creates a mask to remove the diagonal and the upper triangle.
lower_triangle_mask = np.tril(np.ones(df_corr.shape), k=-1).astype("bool")

# Stack all correlations, after applying the mask
df_corr_stacked = df_corr.where(lower_triangle_mask).stack().sort_values()

# Showing the lowest and highest correlations in the correlation matrix
display(df_corr_stacked)
```

Following output will be produced:

```
CNTR_DIST      LONGITUDE      -0.791968
WATER_DIST     LONGITUDE      -0.764256
CNTR_DIST      LATITUDE       -0.717348
age            CNTR_DIST      -0.548287
OCEAN_DIST     LONGITUDE      -0.457477
                                 ...
SPEC_FEAT_VAL  TOT_LVG_AREA    0.506064
CNTR_DIST      WATER_DIST      0.526952
TOT_LVG_AREA   SALE_PRC        0.667301
LONGITUDE      LATITUDE        0.721232
SUBCNTR_DI     CNTR_DIST       0.766387
Length: 136, dtype: float64
```

Quantity Investigation – Outliers

IQR method is used by the Box plot to display data and outliers. But a mathematical formula will be needed to get a list of an identified outlier.

IQR – InterQuartile Range

IQR is the range of values that resides in the middle of scores. In case of skewed distribution and when the median is used instead of the mean to show the central tendency, the suitable measure of variability is the InterQuartile Range, which includes:

- Q1: Lower Quartile Part
- Q2: Median
- Q3: Upper Quartile Part

In other words, IQR is equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles, **IQR = Q3 – Q1.** It is a measure of dispersion similar to standard deviation or variance but is much more robust.

We will be covering outliers in the next exercise. Outliers will be filtered out by keeping only valid values.

Exercise 1.05: Quantity Investigation - Outliers

You are required to analyze the given dataset in terms of its quantity-based features. Load the dataset and import the required libraries. Look for the data cleaning process. Then move forward to perform the Quantity analysis of the data.

First, perform the steps mentioned in Exercise 1.01 and 1.02, 1.03, and 1.04 to cover the features of structure analysis, Quality Investigation, and Content Investigation and then perform the following steps for the computation of Outliers on the given data:

1. For this exercise, Perform the following task for the computation of Outliers though IQR method. IQR method is a measure of dispersion like standard deviation or variance but is much more robust. It is shown here:

```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
# print(df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))
```

```
[27]
df_out = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
df_out.shape
```

Following output will be produced:

```
(10137, 17)
```

Activity 1.01: Loading the "Future Land Use of Miami" Dataset and performing Quality and Structure Investigation on it.

In this activity, you will have to check the working on another dataset. This dataset contains the records of Future Land Usage from Miami. We will be performing Structure and Quality Investigations to test the skill.

The steps are as follows:

1. Open the collab notebook and load the dataset mentioned above. Load it ain the following manner:

```
df =
pd.read_csv('https://raw.githubusercontent.com/fenago/datawrangling/mai
n/Future_Land_Use.csv')
```

2. Import the required libraries. Visualization libraries will be imported for visuals.
3. Perform structure investigation on the data. Show the size of data.
4. Count how many times each data type is present. It will produce the following output:

```
float64    4
int64      2
object     2
dtype: int64
```

5. For each numerical feature, compute a few unique entities.
6. Perform Quality Investigation and check the number of duplicates while ignoring the index feature.
7. Extract column names of all features, except "FID".
8. Drop duplicates. You will see the following output:

```
(914, 8)
```

9. Perform the per feature analysis on the data in terms of missing values.
10. Then display numerical and non-numerical features.

Note: Firstly try to perform it yourself. Then verify it from the solution provided.

To check the solution, you can access it from here:

https://github.com/fenago/datawrangling/blob/main/Chapter%201/Activity_1_01_Structure_Quality_Investigations.ipynb

Conclusion

In this chapter, we have covered all the preprocessing techniques for a dataset. EDA must be performed right after the tasks performed in this chapter. In this way, it becomes easier to deal with a large number of features in a dataset. A proper and detailed EDA takes more time. It is an iterative process. So, a proper start must be taken to avoid vagueness in the upcoming process. This

chapter is revolving around such techniques. It focuses on the EDA basics as well through exercises.

In the next chapter, we will cover EDA in detail including Univariate Analysis, Bivariate Analysis, and multivariate Analysis. Furthermore, the techniques learned in this chapter will be applied in the next chapter. Moreover, these techniques should be applied everywhere before performing the tasks related to Exploratory Data Analysis.