Chapter 1

Introduction

1. Objective

The main intention of this project is revenue prediction on Google's own online advertising platform. The problem at hand is to predict a revenue-related metric for each website traffic record using Google platform dataset. The data sets were sourced from the Kaggle Machine Learning Repository at https://www.kaggle.com/c/machine-learning-battle-mlmath2319/data [1] which is posted as part of Machine Learning Battle -MATH2319. This machine learning analysis project consist of two stages. Phase 1 includes Detailed descriptive statistical analysis of the data using charts, graphs, interactions, etc as appropriate and data pre-processing by dealing with missing values, outliers, data transformation, data aggregation on the Google advertising dataset. The second phase shall include applications of suitable machine learning techniques

This report is divided into 4 parts according to the tasks performed from the initial analysis to the clean dataset preparation. Section 1 explains about the main objective and the description of the data source feature variables and target variables. Section 2 covers data pre-processing and appropriate tools for data wrangling for continuous and categorical features. Section 3 is focused on data exploration of univariate and multivariate features. It also analyses attributes and their inter relationships. The last part presents a brief summary. This report is compiled from Google colab Jupyter Notebook and includes both narratives and the Python codes used for data loading, analysing ,pre-processing, cleansing and visualisation.

2. Data Set

The datasets are part of Machine Learning Battle -MATH 2319 'Revenue Prediction in Online Advertising' [https://www.kaggle.com/c/machine-learning-battle-mlmath2319/overview] at Kaggle. The repository provides 3 datasets . The training set advertising_train.csv , the test set advertising_test.csv and a sample submission file , advertising_sample_submission.csv in the correct format for reference. Phase 1 of this project focus mainly on the Training dataset analysis, exploration and cleansing. Later on, in phase 2 we utilise training and test datasets to apply suitable machine learning technique for the prediction model.

This dataset contains real-world online advertising data where the target feature is a revenue-related metric and the descriptive features are various advertising metrics and characteristics. Each row represents a website traffic record that comes from a specific country, company, and device type combination. The dataset contains 30 days of training data and 5 days of test data. The training data contains about 215K records and the test data contains about 31K records. .Both data sets consist of 20 descriptive features and one target feature. In this project, we worked on training dataset. In Phase II, we would build the appropriate machine learning algorithm from the data set and evaluate their performance using different data validation techniques.

1. Target Feature

Target feature is a revenue related metric of each website from the Google's real-world online advertising data.

y (target feature): revenue-related metric (numeric)

The goal is to predict the value of revenue- related metric for each website traffic record comes from a specific country, company, and device type combination

2. Descriptive Features

Each row represents a website traffic record that comes from a specific country, company, and device type combination. The training and test data consist of 20 >descriptive features. The variable description of the descriptive features are listing below.

• **companyId:** - Company ID of record (categorical)

• **countryId:** - Country ID of record (categorical)

• **deviceType:** - Device type of record (categorical corresponding to desktop, mobile, tablet)

• **day:** -Day of record (integer between 1 (oldest) and 30 for train, 31 and 35 (most recent) for test)

• **dow:** - Day of week of the record (categorical)

• **price1, price2, price3:** - Price combination for the record set by the company (numeric)

• **ad_area:** - area of advertisement (normalized between 0 and 1)

• **ad_ratio:** - ratio of advertisement's length to its width (normalized between 0 and 1)

• **requests, impression, cpc, ctr, viewability:** - Various metrics related to the record (numeric)

• **ratio1, …, ratio5:** - Ratio characteristics related to the record (each normalized between 0 and 1)

The currency features are in US Dollars.

Most of the descriptive features are self-explanatory, except, impression, cpc, ctr .Explanations for these metrics are listed below[2].

•**Impressions :**-The number of times your ad has been shown

•**ctr: - Click Trhough Rate :** - is the number of Clicks an Ad receives divided by the number of times the Ad is shown (Impressions) expressed as a percentage.

•**cpc:-Cost Per Click :** -CPC is the average amount that you have been charged for each click.

Chapter 2

Data Pre-Processing

1. Preliminaries

We read the training dataset with the help of Jupiter Notebook. For the analysis purpose we have stored the dataset into the google drive and loaded data using Google Colab platform. The dataset contains attribute names and and renamed some of the column names for better readability. Sample of the loaded dataset is

```
#Import the required libraries into the work space
import pandas as pd
import codecs
import numpy as np
import csv
import math
from io import StringIO



#check the current working directory
pwd
```

```
#Mounting Google Drive locally

from google.colab import drive
drive.mount('/content/gdrive')

#Open the training dataset using f.read() function

with open('/content/gdrive/My Drive/advertising_train.csv', 'r') as f:
 data = f.read()

#Store the dataset into a variable

d_adv = pd.read_csv(StringIO(data.replace('”', '"')), sep=',', encoding='utf-8',
                    quoting=csv.QUOTE_NONNUMERIC, dtype={'companyId':'int64',
 'countryId':'int64', 'deviceType':'category', 'day':'int64','dow':'category',
'price1':'float64','price2':'float64','price3':'float64',
'ad_area':'float64', 'ad_ratio':'float64','requests':'int64', 'impression':
  'int64', 'cpc':'float64','ctr':'float64','viewability':
'float64','ratio1':'float64','ratio2':'float64','ratio3':'float64','ratio4':
 'float64','ratio5':'float64','y':'float64' })


#Change some of the the column names for better understanding

col_Names = ["company_id","country_id","device_type","day","day_of_week",
             "price_1",
             "price_2","price_3","ad_area","ad_ratio","requests","impression",
             "c_p_c","c_t_r","viewability","ratio_1","ratio_2","ratio_3",
             "ratio_4","ratio_5","revenue_related_metric"]
d_adv.columns=col_Names
```

```
#check the loaded dataset samples

d_adv.head()
```

1. Data Cleaning and Transformation

Initially step of the data cleansing process is to validate the dataset with the original sample.We confirmed the dimension, attribute names and the data types of the dataset.

```
#Display the dimension and the column names

print(f"Dimension of the data set is {d_adv.shape}\n")
print(f"Data Types are: ")
print(d_adv.dtypes)
```

We checked if there any index values present in the data frame .This dataset is loaded with a well defined index values.

```
#check the index values
d_adv.index.values
```

Secondly, we searched for missing values in any of the attributes using the below code chunk.From the result, it's obvious that this dataset does not contains any missing values.

```
print(f"\nNumber of missing value for each feature:")
print(d_adv.isnull().sum())
```

Finally, we checked the summary statistics of the descriptive and categorical features separately for the later analysis and verified if there are any abnormalities in the dataset from the normal behaviour. We

includedexcluded the ID values here .Adetailed analysis of each descriptive features will perform in the following sections.

Double-click (or enter) to edit

```
from IPython.display import display,HTML

#remove ID values
list_drop = ['company_id', 'country_id']
d=d_adv.copy()
d.drop(list_drop, axis=1, inplace=True)

#continuous attributes int
display(HTML('<b>Table 1: Summary of integer features</b>'))
display(d.describe(include = 'int64'))
```

```
#continuous attributes float
display(HTML('<b>Table 1: Summary of float features</b>'))
display(d_adv.describe(include = 'float64'))
```

```
#categorical attributes
display(HTML('<b>Table 2: Summary of categorical features</b>'))
display(d_adv.describe(include = 'category'))
```

We went deeper to analyse the descriptive features and as a next step we checked the frequency table of the continuous features and descriptive features separately.

1. **Continuous Features**

```
print("frequency table of integer values \n'")

for col in d_adv.columns:
    if (d_adv[col].dtype.name == 'int64'):
        print(d_adv[col].value_counts(), '\n')
```

```
print("Frequency table of float values \n'")

for col in d_adv.columns:
    if (d_adv[col].dtype.name == 'float64'):
        print(d_adv[col].value_counts(), '\n')
```

Company ID

43 134655

159 44877

95 19731

40 11744

126 1844

157 1277

Name: company_id, dtype: int64

From this frequency table we can observe that the data is skewed to one direction. That is for the company Id 126 and 157 the value count is very low and the others pretty high as compared with these IDs.

Country ID ,Day, Request and Impression

These integer values seems to be fine during the initial frequency table analysis and more detailed descriptions will be checking in the visualisation process.

All the other float values are checked thoroughly from the frequency table and as a common trend, we could find that when request value is zero all the other metrics such as impression, cpc, ctr, viewability and ratio values are also zero. We need to analyse more with the visualisation tools on these strange behaviour in this dataset and have to drop these datasets if they are not providing any valid information.

1. **Categorical Features**

The categorical features in the dataset are device type and day of the week. There were no white spaces in these characters. Even though, there is no white space we have added the code to trim the white spaces from the categorical features as below.

device type

2 94827

1 75081

3 42107

5 2113

Name: device_type, dtype: int64

Since device type of record (categorical corresponding to desktop, mobile, tablet) we are ignoring the category 5 for further analysis. We could see that the count very low for the category 5 as compared with category 1 ,2 and 3. We will remove these rows from the dataset after confirming this by drawing the descriptive visualisations.

Day of week

Saturday 35200

Sunday 33882

Tuesday 29906

Monday 29836

Thursday 29387

Wednesday 28215

Friday 27702

Name: day_of_week, dtype: int64

Day of the week is a categorical feature and its distributed across all the 7 days almost equally. Even though the count for Friday is less, its is not violating the normalisation criteria for data analysis. This categorical feature seems to be perfect for the further analysis

```
for col in d_adv.columns:
    if (d_adv[col].dtype.name in ['category', 'str', 'object']):
        # 4. Check for extra whitespaces and remove them
        d_adv[col] = d_adv[col].str.strip()
        # 5. Cast text to lower-case
        d_adv[col] = d_adv[col].str.lower()
```

```
print("frequency table of categorical values \n'")
for col in d_adv.columns:
    if (d_adv[col].dtype.name == 'object'):
        print(d_adv[col].value_counts(), '\n')
```

Chapter 3

Data Exploration

Visual data analysis is an important first step in any advanced analytics project .Data exploration is a recommended first step in any analysis.The visual approach also highlights important aspects of data sets. It can also illuminate correlations between two variables

1. Univariate Visualisations

1. Categorical Features

For convenience, we defined two functions named BarPlot(x) and BoxHistogramPlot(x) for categorical and numerical features respectively. For given an input categorical column x, BarPlot(x) returns a bar chart with percentage on top of each bar. A bar chart is useful to present the proportions by categories. For given an input numerical column x, BoxHistogramPlot(x) plots a histogram and a box plot. A histogram is useful to visualize the shape of the underlying distribution whereas A box plot tells the range of the attribute and helps detect any outliers. The following chunk codes show how these functions were defined using the numpy library and the matplotlib library[3].

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)

def BarPlot(x):
  total = float(len(d_adv))
  ax = d_adv[x].value_counts(normalize = True).plot( kind = "bar", alpha = 0.5)

def BoxHistogramPlot(x):
  f, (ax_box, ax_hist) = plt.subplots(2, sharex=True, gridspec_kw={"height_ratios": (.15, .85

  sns.boxplot(x, ax=ax_box)
  sns.distplot(x, ax=ax_hist)
  ax_box.set(yticks=[])
  sns.despine(ax=ax_hist)
  sns.despine(ax=ax_box, left=True)
  plt.show()
```

We plotted bar chart for all the categorical features present in the dataset. Those include Company id ,Day, Device type and Day of the week

```
#initialize figure labelling
i=1
for col in ['company_id','day','device_type', 'day_of_week']:
  plt.figure(figsize=(8,4))
  plt.title("Figure " + str(i) + ": Bar Chart of " + col, fontsize = 12)
  BarPlot(col)
  plt.show()
  i = i + 1
```

Figure 1 depicts the bar chart distribution of company ID. From the graph we could conclude that more than 50% of the data is from the company with Company ID 43 and followed by Company 159. Companies having Id 126 and 157 contribute really less in the dataset. This is the serious issue to be considered because our modelling may biased and more favourable to the companies with higher population in rows.

Figure 2 describes the bar chart of the day. From the graph it's clear that almost all days have the contribution of .030 to .035 and day 2 has the lowest which is less than 0.020.This descriptive feature is suitable for data analysis purpose because it is distributed almost equally among all the days.

This figure describers the distribution of device type among the companies. From this dataset its clear that only 3 of the features are relevant and the device type 5 can be consider as outlier. Since the row count of device type 5 is less as compared to others, we could remove these features from the dataset. The below code represents the bar chart without the device type with value 5

```
#remove device type 5

d_adv = d_adv[d_adv['device_type'] != "5"]

#Barplot
d_adv['device_type'].value_counts().plot(kind='bar')
plt.title('COMPANIES\n')
plt.xlabel('Device Type')
plt.ylabel('frequency')
plt.legend(['Device Type frequency '])
plt.show()
```

Figure 4 shows the distribution of day of week in companies. From this dataset its clear that most of the days have values around 14 to 16 and its distributed equally among the dataset. Since this values spread

among the dataset almost equally, we can proceed with these descriptive feature without any change.

1. Continuous features

The BoxHistogramPlot function was applied to all numeric variables.The below code chunk shows the coding for the continuous variable.

```
d_adv[d_adv.company_id.isnull()].head()
```

```
#Visualisation for price features

for col in ['price_1','price_2', 'price_3']:
    plt.suptitle("Figure " + str(i) + ": Histogram and Box Plot of " + col)
    BoxHistogramPlot(d_adv[col])
    plt.show()
    i = 1 + i
```

When we analyse the box plot and histogram of price attributes, we realised that price attributes with value 0 has the maximus count in all the 3 datasets which is already listed out when we did data analysis in the initial step. Since these row counts is more than 30 % of the original dataset, we cannot ignore these column values. So, we replaced the price values having 0 values with mean values from the dataset.

```
#Replace with mean
price1_mean = d_adv[ d_adv.price_1 != 0 ].mean()
price2_mean = d_adv[ d_adv.price_2 != 0 ].mean()
price2_mean = d_adv[ d_adv.price_3 != 0 ].mean()

d_adv.loc[ d_adv.price_1 == 0, "price_1" ] = price1_mean
d_adv.loc[ d_adv.price_2 == 0, "price_2" ] = price2_mean
d_adv.loc[ d_adv.price_3 == 0, "price_3" ] = price3_mean
```

```
#Visualisation for price features after replacement

for col in ['price_1','price_2', 'price_3']:
    plt.suptitle("Figure " + str(i) + ": Histogram and Box Plot of " + col)
    BoxHistogramPlot(d_adv[col])
    plt.show()
    i = 1 + i
```

```
#d_adv.dropna(axis=0, how='any', inplace=True)
```

```
##Exploratory Data Analysis
```

```
#Visualization for 22  ? Id?
```

```
import matplotlib.pyplot as plt
```

```
d_adv['case_id'].plot(kind='box')
plt.title('Unique Case_ID\n')
plt.xlabel('case_id')
plt.ylabel('density')
plt.legend(['Unique Case_ID'])
plt.show()
```

```
d_adv['case_id'].plot(kind='density')
plt.title('Unique Case_ID\n')
plt.xlabel('case_id')
plt.ylabel('density')
plt.legend(['Unique Case_ID'])
plt.show()
```

```
d_adv['company_id'].value_counts().plot(kind='bar')
plt.title('COMPANIES\n')
plt.xlabel('companyId')
plt.ylabel('frequency')
plt.legend(['COMPANY IDs frequency '])
plt.show()
```

```
d_adv['country_id'].value_counts().plot(kind='pie')
plt.title('COUNTRIES\n')
plt.xlabel('countryId')
plt.ylabel('frequency')
plt.legend(['COUNTRY frequency '])
plt.show()


d_adv['device_type'].value_counts().plot(kind='bar')
plt.title('DEVICE TYPE\n')
plt.xlabel('deviceType')
plt.ylabel('frequency')
plt.legend(['Device Types frequency \n'])
plt.show()



d_adv['day'].value_counts().plot(kind='bar')
plt.title('DAY')
plt.xlabel('day')
plt.ylabel('frequency')
plt.legend([' DAY FREQUENCY '])
plt.show()



d_adv['day_of_week'].value_counts().plot(kind='bar')
plt.title('DAY OF WEEK \n')
plt.xlabel('Day of Week')
plt.ylabel('frequency')
plt.legend([' DAY OF WEEK FREQUENCY '])
plt.show()



d_adv['price_1'].plot(kind='density')
plt.title('price1 \n')
plt.xlabel('price1')
plt.ylabel('density')
plt.legend(['PRICE-1'])
plt.show()



d_adv['price_2'].plot(kind='box')
plt.title('price2 \n')
plt.xlabel('price2')
plt.ylabel('density')
plt.legend(['PRICE-2'])
plt.show()



d_adv['price_3'].plot(kind='box')
plt.title('price3 \n')
plt.xlabel('price3')
plt.ylabel('density')
plt.legend(['PRICE-3'])
plt.show()



d_adv['ad_area'].plot(kind='box')
plt.title('ad_area \n')
plt.xlabel('ad_area')
plt.ylabel('density')
plt.legend(['ad_area'])
plt.show()
```

```
d_adv['ad_ratio'].plot(kind='density')
plt.title('AD RATIO \n')
plt.xlabel('ad_ratio')
plt.ylabel('density')
plt.legend(['Ad Ratio'])
plt.show()
```

```
d_adv['ad_ratio'].plot(kind='box')
plt.title('AD RATIO \n')
plt.xlabel('ad_ratio')
plt.ylabel('density')
plt.legend(['Ad Ratio'])
plt.show()
```

```
d_adv['requests'].plot(kind='density')
plt.title('Ad REQUESTS \n')
plt.xlabel('requests')
plt.ylabel('density')
plt.legend(['Ad Requests'])
plt.show()
```

```
d_adv['requests'].plot(kind='box')
plt.title('Ad REQUESTS \n')
plt.xlabel('requests')
plt.ylabel('density')
plt.legend(['Ad Requests'])
plt.show()
```

```
d_adv['impression'].plot(kind='density')
plt.title('IMPRESSION \n')
plt.xlabel('impression')
plt.ylabel('density')
plt.legend(['Impression'])
plt.show()
```

```
d_adv['impression'].plot(kind='box')
plt.title('IMPRESSION \n')
plt.xlabel('impression')
plt.ylabel('density')
plt.legend(['Impression'])
plt.show()
```

```
d_adv['c_p_c'].plot(kind='box')
plt.title('CPC \n')
plt.xlabel('cpc')
plt.ylabel('density')
plt.legend(['cpc'])
plt.show()
```

```
d_adv['cpc'].plot(kind='density')
plt.title('CPC \n')
plt.xlabel('cpc')
plt.ylabel('density')
plt.legend(['cpc'])
plt.show()
```

```
d_adv['c_t_r'].plot(kind='density')
plt.title('CTR \n')
plt.xlabel('ctr')
plt.ylabel('density')
plt.legend(['ctr'])
plt.show()
```

```
d_adv['c_t_r'].plot(kind='box')
plt.title('CTR \n')
plt.xlabel('ctr')
plt.ylabel('density')
plt.legend(['ctr'])
plt.show()
```

```
d_adv['viewability'].plot(kind='density')
plt.title('VIEWABILITY \n')
plt.xlabel('viewability')
plt.ylabel('density')
plt.legend(['viewability'])
plt.show()
```

```
d_adv['viewability'].plot(kind='box')
plt.title('VIEWABILITY \n')
plt.xlabel('viewability')
plt.ylabel('density')
plt.legend(['viewability'])
plt.show()
```

```
d_adv['ratio_1'].plot(kind='density')
plt.title('RATIO1 \n')
plt.xlabel('ratio1')
plt.ylabel('density')
plt.legend(['ratio1'])
plt.show()
```

```
d_adv['ratio_2'].plot(kind='density')
plt.title('RATIO 2 \n')
plt.xlabel('ratio2')
plt.ylabel('density')
plt.legend(['ratio2'])
plt.show()
```

```
d_adv['ratio_3'].plot(kind='density')
plt.title('RATIO3 \n')
plt.xlabel('ratio3')
plt.ylabel('density')
plt.legend(['ratio3'])
plt.show()
```

```
d_adv['ratio_4'].plot(kind='density')
plt.title('RATIO4 \n')
plt.xlabel('ratio4')
plt.ylabel('density')
plt.legend(['ratio4'])
plt.show()
```

```
d_adv['ratio_5'].plot(kind='density')
plt.title('RATIO5 \n')
plt.xlabel('ratio5')
plt.ylabel('density')
plt.legend(['ratio5'])
plt.show()
```

```
d_adv['revenue_related_metric'].plot(kind='density')
plt.title('Y \n')
plt.xlabel('y')
plt.ylabel('density')
plt.legend(['y'])
plt.show()
```

```
d_adv['revenue_related_metric'].plot(kind='hist')
plt.title('Y \n')
plt.xlabel('y')
plt.ylabel('density')
plt.legend(['y'])
plt.show()
```

```
d_adv['revenue_related_metric'].plot(kind='box')
plt.title('Y \n')
plt.xlabel('y')
plt.ylabel('density')
plt.legend(['y'])
plt.show()
```

```
# Exploring the relationship between columns
```

```
d_adv.plot(kind='scatter', x='company_id', y='country_id')
plt.title('Relation between CompanyId and CountryId\n')
plt.xlabel('companyId')
plt.ylabel('countryId')
plt.legend(['CompanyId/CountryId'])
plt.show()
```

```
d_adv.plot(kind='scatter', x='country_id', y='company_id')
plt.title('Relation between CountryId and CompanyIdId Descriptive Features\n')
plt.xlabel('CountryId')
plt.ylabel('CompanyIdId')
plt.legend(['CountryId/CompanyIdId'])
plt.show()
```

```
# It seems that these two features are dependent to each other, some companies are related to
```

```
d_adv.plot(kind='scatter', x='company_id', y='day')
plt.title('Relation between CompanyId and day\n')
plt.xlabel('companyId')
plt.ylabel('day')
plt.legend(['CompanyId/day'])
plt.show()
```

```
d_adv.plot(kind='scatter', x='company_id', y='price_1')
plt.title('Relation between CompanyId and Price1\n')
plt.xlabel('companyId')
plt.ylabel('price1')
plt.legend(['CompanyId/price1'])
plt.show()
```

```
d_adv.plot(kind='scatter', x='company_id', y='ad_area')
plt.title('Relation between CompanyId and day\n')
plt.xlabel('companyId')
plt.ylabel('day')
plt.legend(['CompanyId/day'])
plt.show()
```