

**ANL252**

**Python for Data Analytics**

# **End-of-Course Assessment**

**January 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Ang Cheer Neng** | **K2270972** |

**Tutorial Group: ­­­­­­­­­­­­ T 05**

**Instructor’s Name: Dr. Munish Kumar**

**Submission Date: 05/03/2023**

Q1

The variables that contain missing values are “Claim\_ID” and “Actual”.

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

# to read csv file

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# to find the variables with missing values

insurance\_data.isnull().sum(axis **=** 0**/**1)

Claim\_ID 5

Policy\_No 0

Name 0

Planned 0

Actual 1677

Created 0

Amount 0

Paid 0

Category 0

Terms 0

Region 0

Type 0

dtype: int64

Q2

The missing data were treated by deleting the rows containing the missing data. This is because the rows with missing data will not be able to be used when the data needs to be analysed or categorised. It can cause inaccuracy in the result if the missing data were not removed.

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

# to read csv file

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# to delete the rows with missing value

updated\_insurance\_data **=** insurance\_data.dropna(axis**=**0, how **=**'any')

updated\_insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24085** | 3.960616e+09 | 240104423 | Mary Taylor | 8/3/2022 | 6/3/2022 0:00 | 20220303 | 6126.018 | Yes | AT | CB91 | FVS | O001 |
| **24086** | 3.960616e+09 | 240104423 | Sarah Holland | 7/3/2022 | 9/3/2022 0:00 | 20220302 | 6288.599 | Yes | AT | CB91 | FVS | O001 |
| **24087** | 3.960616e+09 | 240104429 | Sean Foster | 15/3/2022 | 23/3/2022 0:00 | 20220310 | 3164.472 | Yes | AT | CB91 | FVS | O001 |
| **24088** | 3.960616e+09 | 240104429 | Tammy Duncan | 8/3/2022 | 23/3/2022 0:00 | 20220304 | 1150.452 | Yes | AT | CB91 | FVS | O001 |
| **24090** | 3.960617e+09 | 240105686 | Mr. Robert Rivera | 10/3/2022 | 12/3/2022 0:00 | 20220305 | 603.68 | Yes | AT | CB91 | FVS | O001 |

22531 rows × 12 columns

# to find the variables with missing values

updated\_insurance\_data.isnull().sum(axis **=** 0**/**1)

Claim\_ID 0

Policy\_No 0

Name 0

Planned 0

Actual 0

Created 0

Amount 0

Paid 0

Category 0

Terms 0

Region 0

Type 0

dtype: int64

Q3

The first data preparation task is forward filling of the missing data. This means to generate the next data according to the previous data. Any missing data will be filled according to the previous rows. This is a good way to treat as it follows the available data to input the missing ones so that the data will become complete and can be utilised for analysing again.

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

# to read csv file

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# to fill the missing rows with data from previous data

second\_insurance\_data **=** insurance\_data.ffill(axis **=** 0**/**1)

second\_insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | 3.960634e+09 | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | 3.960634e+09 | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | 3.960634e+09 | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | 3.960634e+09 | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | 3.960634e+09 | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# to check the rows if any missing values

second\_insurance\_data.isnull().sum(axis **=** 0**/**1)

Claim\_ID 0

Policy\_No 0

Name 0

Planned 0

Actual 0

Created 0

Amount 0

Paid 0

Category 0

Terms 0

Region 0

Type 0

dtype: int64

The second way to treat the data is to sort the data in ascending or descending order. This can help to analyse the data according to which columns is sorted. Referring to the results below, the amount of the policy has been sorted in descending order. The data now shows which policy has the highest or lowest price and which policies are the more expensive ones. It can also be analysed why there are policies that are more expensive. Hence, this is a good treatment of the data for further analysis.

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

# to read csv file

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# to sort the column “Amount” from largest to smallest amount

insurance\_data.sort\_values(

by **=** ["Amount"], ascending**=False**)

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **14031** | 2.929952e+09 | 300920735 | Joseph Brady | 15/10/2021 | 15/10/2021 0:00 | 20211010 | 9993.246 | Yes | AT | AD23 | LOC | L001 |
| **17009** | 2.930208e+09 | 300769623 | David Anthony | 18/12/2021 | 16/12/2021 0:00 | 20211213 | 999.772 | Yes | AT | AB27 | LOC | L001 |
| **21682** | 2.930813e+09 | 300416837 | Tracy Hicks | 26/4/2022 | NaN | 20220421 | 999.318 | No | AT | DA17 | LOC | L001 |
| **14801** | 2.930022e+09 | 300722444 | Kayla Morales | 25/11/2021 | 6/12/2021 0:00 | 20211120 | 999.24 | Yes | AT | EC05 | LOC | L001 |
| **22511** | 3.960531e+09 | 240104225 | Crystal Rogers | 16/3/2021 | 11/3/2021 0:00 | 20210311 | 999.093 | Yes | AT | CB91 | FVS | O001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **17955** | 2.930342e+09 | 300726979 | Mark Griffin | 17/1/2022 | 21/1/2022 0:00 | 20220112 | 100.101 | Yes | AT | AD23 | LOC | L001 |
| **7214** | 2.929349e+09 | 300726979 | Denise Higgins | 30/5/2021 | 30/5/2021 0:00 | 20210525 | 100.089 | Yes | AT | AD23 | LOC | L001 |
| **16989** | 2.930207e+09 | 300769623 | Robert Martin | 16/12/2021 | 13/12/2021 0:00 | 20211211 | 100.069 | Yes | AT | AB27 | LOC | L001 |
| **4201** | 2.929017e+09 | 300769623 | Paul Chase | 8/4/2021 | 4/4/2021 0:00 | 20210403 | 100.05 | Yes | AT | AB27 | LOC | L001 |
| **9468** | 2.929572e+09 | 300705742 | Kristopher Porter | 7/8/2021 | 19/7/2021 0:00 | 20210801 | 100.013 | Yes | AT | EB64 | LOC | L001 |

24213 rows × 12 columns

The third way to treat the data is to reduce the number of columns in the data. This is so that the columns that are less important can be removed and focus is shifted to the columns that are needed for further analysis. As per the results below, all the other columns are removed other than the amount of the policy and the category of the policy. The remaining columns are used to analyse how much are usually spent on which category of policy. Hence, this treatment of data helps to further analyse the data.

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

# to read csv file

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# to delete the columns other than “Amount” and “Category”

insurance\_data.rename(columns **=** {'Claim\_ID' : 'A', 'Policy\_No' : 'B','Name' : 'C', 'Planned': 'D', 'Actual' : 'E', 'Created': 'F', 'Amount' : 'G', 'Paid':'H', 'Category' : 'I', 'Terms' : 'J', 'Region' : 'K', 'Type':'L'}, inplace **=** **True**)

insurance\_data

|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

third\_insurance\_data **=** insurance\_data.drop(['A','B','C','D','E','F','H','J','K','L'], axis**=**1)

third\_insurance\_data

|  | **G** | **I** |
| --- | --- | --- |
| **0** | 3072.349 | AT |
| **1** | 910.944 | AT |
| **2** | 567.936 | AT |
| **3** | 181.651 | AT |
| **4** | 238.74 | AT |
| **...** | ... | ... |
| **24208** | 188.4 | XT |
| **24209** | 460.8 | XT |
| **24210** | 120.735 | XT |
| **24211** | 591.12 | XT |
| **24212** | 837 | XT |

24213 rows × 2 columns

third\_insurance\_data.rename(columns **=** {'G': "Amount",'I': "Category"}, inplace **=** **True**)

third\_insurance\_data

|  | **Amount** | **Category** |
| --- | --- | --- |
| **0** | 3072.349 | AT |
| **1** | 910.944 | AT |
| **2** | 567.936 | AT |
| **3** | 181.651 | AT |
| **4** | 238.74 | AT |
| **...** | ... | ... |
| **24208** | 188.4 | XT |
| **24209** | 460.8 | XT |
| **24210** | 120.735 | XT |
| **24211** | 591.12 | XT |
| **24212** | 837 | XT |

24213 rows × 2 columns

Q4.

The first insight of the corporate claims is that the type of claim that has the highest amount paid is L001. The claim with the lowest amount paid is L004. This shows that some type of claims are more popular than other type of claims.

The second insight of the data is there is a difference between the planned date and the actual date of the claim settlement. The illustration shows the number of times of the difference in the planned and actual date of claim settlement. From the illustration, we can infer that the difference that occurred the most times is 0 days where it happened about 5000 times. There are a few outliers where it happened once there was a -272 days difference in the planned and actual date. This shows that usually there will seldom be a lag of claim settlement and people get their settlement without delay.

The third insight is region LOC has a much higher total amount of payout as compared to region FVS. This means that more people in LOC filed for a claim to their insurance and gotten a payout of a certain amount and less people in FVS needed to file a claim. As per the illustration, we can infer that the people in LOC filed more claims than the people in FVS and also the total amount of payout was about 6 times more than FVS.

Q5

Firstly, the columns of the data was renamed into alphabets and removed the columns other than the “Planned” and “Actual” columns. Secondly, the date in the “Planned” and “Actual” were converted into datetime format. Thirdly, the datetime for both “Planned” and “Actual” were converted from float to integers. Then, the “Planned” data are assigned to x-axis and “Actual” data are assigned to y-axis. After which, the columns were combined before plotting the linear regression model.

**import** pandas **as** pd

**import** sklearn

**from** sklearn.linear\_model **import** LinearRegression

**import** matplotlib.pyplot **as** plt

**from** scipy **import** stats

**import** numpy **as** np

#to read csv file

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# rename columns to alphabets

insurance\_data.rename(columns **=** {'Claim\_ID' : 'A', 'Policy\_No' : 'B','Name' : 'C', 'Planned': 'D', 'Actual' : 'E', 'Created': 'F', 'Amount' : 'G', 'Paid':'H', 'Category' : 'I', 'Terms' : 'J', 'Region' : 'K', 'Type':'L'}, inplace **=** **True**)

insurance\_data

|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 5/2/2021 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 18/1/2021 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 15/1/2021 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 5/2/2021 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 30/3/2021 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 18/3/2021 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 21/3/2021 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 1/4/2021 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 4/4/2021 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

24213 rows × 12 columns

# delete columns other than D and E which are planned and actual date

fifth\_insurance\_data **=** insurance\_data.drop(['A','B','C','F','G','H','I','J','K','L'], axis**=**1)

fifth\_insurance\_data

|  | **D** | **E** |
| --- | --- | --- |
| **0** | 17/1/2021 | 18/1/2021 0:00 |
| **1** | 5/2/2021 | 16/1/2021 0:00 |
| **2** | 18/1/2021 | 14/1/2021 0:00 |
| **3** | 15/1/2021 | 18/1/2021 0:00 |
| **4** | 5/2/2021 | 8/2/2021 0:00 |
| **...** | ... | ... |
| **24208** | 30/3/2021 | 27/12/2021 0:00 |
| **24209** | 18/3/2021 | 31/5/2021 0:00 |
| **24210** | 21/3/2021 | 23/5/2021 0:00 |
| **24211** | 1/4/2021 | 22/5/2021 0:00 |
| **24212** | 4/4/2021 | 8/5/2021 0:00 |

24213 rows × 2 columns

# to convert dates in D column to datetime

fifth\_insurance\_data['D'] **=** pd.to\_datetime(fifth\_insurance\_data['D'])

fifth\_insurance\_data

|  | **D** | **E** |
| --- | --- | --- |
| **0** | 2021-01-17 | 18/1/2021 0:00 |
| **1** | 2021-05-02 | 16/1/2021 0:00 |
| **2** | 2021-01-18 | 14/1/2021 0:00 |
| **3** | 2021-01-15 | 18/1/2021 0:00 |
| **4** | 2021-05-02 | 8/2/2021 0:00 |
| **...** | ... | ... |
| **24208** | 2021-03-30 | 27/12/2021 0:00 |
| **24209** | 2021-03-18 | 31/5/2021 0:00 |
| **24210** | 2021-03-21 | 23/5/2021 0:00 |
| **24211** | 2021-01-04 | 22/5/2021 0:00 |
| **24212** | 2021-04-04 | 8/5/2021 0:00 |

24213 rows × 2 columns

# to convert dates in E column to datetime

fifth\_insurance\_data['E'] **=** pd.to\_datetime(fifth\_insurance\_data['E'])

fifth\_insurance\_data

|  | **D** | **E** |
| --- | --- | --- |
| **0** | 2021-01-17 | 2021-01-18 |
| **1** | 2021-05-02 | 2021-01-16 |
| **2** | 2021-01-18 | 2021-01-14 |
| **3** | 2021-01-15 | 2021-01-18 |
| **4** | 2021-05-02 | 2021-08-02 |
| **...** | ... | ... |
| **24208** | 2021-03-30 | 2021-12-27 |
| **24209** | 2021-03-18 | 2021-05-31 |
| **24210** | 2021-03-21 | 2021-05-23 |
| **24211** | 2021-01-04 | 2021-05-22 |
| **24212** | 2021-04-04 | 2021-08-05 |

24213 rows × 2 columns

# delete the rows with missing values

updated\_fifth\_insurance\_data **=** fifth\_insurance\_data.dropna(axis**=**0, how **=**'any')

updated\_fifth\_insurance\_data

|  | **D** | **E** |
| --- | --- | --- |
| **0** | 2021-01-17 | 2021-01-18 |
| **1** | 2021-05-02 | 2021-01-16 |
| **2** | 2021-01-18 | 2021-01-14 |
| **3** | 2021-01-15 | 2021-01-18 |
| **4** | 2021-05-02 | 2021-08-02 |
| **...** | ... | ... |
| **24208** | 2021-03-30 | 2021-12-27 |
| **24209** | 2021-03-18 | 2021-05-31 |
| **24210** | 2021-03-21 | 2021-05-23 |
| **24211** | 2021-01-04 | 2021-05-22 |
| **24212** | 2021-04-04 | 2021-08-05 |

22536 rows × 2 columns

# convert datetime from column “D” to float in “D\_delta”

updated\_fifth\_insurance\_data['D\_delta'] **=** (updated\_fifth\_insurance\_data['D'] **-** updated\_fifth\_insurance\_data['D'].min())**/**np.timedelta64(1,'D')

updated\_fifth\_insurance\_data

|  | **D** | **E** | **D\_delta** |
| --- | --- | --- | --- |
| **0** | 2021-01-17 | 2021-01-18 | 17.0 |
| **1** | 2021-05-02 | 2021-01-16 | 122.0 |
| **2** | 2021-01-18 | 2021-01-14 | 18.0 |
| **3** | 2021-01-15 | 2021-01-18 | 15.0 |
| **4** | 2021-05-02 | 2021-08-02 | 122.0 |
| **...** | ... | ... | ... |
| **24208** | 2021-03-30 | 2021-12-27 | 89.0 |
| **24209** | 2021-03-18 | 2021-05-31 | 77.0 |
| **24210** | 2021-03-21 | 2021-05-23 | 80.0 |
| **24211** | 2021-01-04 | 2021-05-22 | 4.0 |
| **24212** | 2021-04-04 | 2021-08-05 | 94.0 |

22536 rows × 3 columns

# convert datetime from column “E” to float in “E\_delta”

updated\_fifth\_insurance\_data['E\_delta'] **=** (updated\_fifth\_insurance\_data['E'] **-** updated\_fifth\_insurance\_data['E'].min())**/**np.timedelta64(1,'D')

updated\_fifth\_insurance\_data

|  | **D** | **E** | **D\_delta** | **E\_delta** |
| --- | --- | --- | --- | --- |
| **0** | 2021-01-17 | 2021-01-18 | 17.0 | 16.0 |
| **1** | 2021-05-02 | 2021-01-16 | 122.0 | 14.0 |
| **2** | 2021-01-18 | 2021-01-14 | 18.0 | 12.0 |
| **3** | 2021-01-15 | 2021-01-18 | 15.0 | 16.0 |
| **4** | 2021-05-02 | 2021-08-02 | 122.0 | 212.0 |
| **...** | ... | ... | ... | ... |
| **24208** | 2021-03-30 | 2021-12-27 | 89.0 | 359.0 |
| **24209** | 2021-03-18 | 2021-05-31 | 77.0 | 149.0 |
| **24210** | 2021-03-21 | 2021-05-23 | 80.0 | 141.0 |
| **24211** | 2021-01-04 | 2021-05-22 | 4.0 | 140.0 |
| **24212** | 2021-04-04 | 2021-08-05 | 94.0 | 215.0 |

22536 rows × 4 columns

# to delete the columns “D” and “E”

sixth\_insurance\_data **=** updated\_fifth\_insurance\_data.drop(['D','E'], axis**=**1)

sixth\_insurance\_data

|  | **D\_delta** | **E\_delta** |
| --- | --- | --- |
| **0** | 17.0 | 16.0 |
| **1** | 122.0 | 14.0 |
| **2** | 18.0 | 12.0 |
| **3** | 15.0 | 16.0 |
| **4** | 122.0 | 212.0 |
| **...** | ... | ... |
| **24208** | 89.0 | 359.0 |
| **24209** | 77.0 | 149.0 |
| **24210** | 80.0 | 141.0 |
| **24211** | 4.0 | 140.0 |
| **24212** | 94.0 | 215.0 |

22536 rows × 2 columns

# to convert float in “D\_delta” to integer

sixth\_insurance\_data **=** sixth\_insurance\_data.astype({'D\_delta':'int'})

sixth\_insurance\_data

|  | **D\_delta** | **E\_delta** |
| --- | --- | --- |
| **0** | 17 | 16.0 |
| **1** | 122 | 14.0 |
| **2** | 18 | 12.0 |
| **3** | 15 | 16.0 |
| **4** | 122 | 212.0 |
| **...** | ... | ... |
| **24208** | 89 | 359.0 |
| **24209** | 77 | 149.0 |
| **24210** | 80 | 141.0 |
| **24211** | 4 | 140.0 |
| **24212** | 94 | 215.0 |

22536 rows × 2 columns

# to convert float in “E\_delta” to integer

sixth\_insurance\_data **=** sixth\_insurance\_data.astype({'E\_delta':'int'})

sixth\_insurance\_data

|  | **D\_delta** | **E\_delta** |
| --- | --- | --- |
| **0** | 17 | 16 |
| **1** | 122 | 14 |
| **2** | 18 | 12 |
| **3** | 15 | 16 |
| **4** | 122 | 212 |
| **...** | ... | ... |
| **24208** | 89 | 359 |
| **24209** | 77 | 149 |
| **24210** | 80 | 141 |
| **24211** | 4 | 140 |
| **24212** | 94 | 215 |

22536 rows × 2 columns

**import** matplotlib.pyplot **as** plt

**from** scipy **import** stats

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_absolute\_error, mean\_squared\_error, r2\_score

# to assign x-axis and y-axis

x **=** sixth\_insurance\_data['D\_delta']

y **=** sixth\_insurance\_data['E\_delta']

**from** sklearn **import** preprocessing

**from** sklearn **import** linear\_model

array\_list\_1 **=** np.array([sixth\_insurance\_data['D\_delta']])

arr1 **=** array\_list\_1

resultant\_string\_1 = np.reshape(arr1, (22536, -1))

array\_list\_2 **=** np.array([sixth\_insurance\_data['E\_delta']])

arr2 **=** array\_list\_2

resultant\_string\_2 **=** np.reshape(arr2, (22536, **-**1))

# to rename x-axis and y-axis

x **=** resultant\_string\_1

y **=** resultant\_string\_2

# to combine x-axis and y-axis into array

end\_string **=** np.concatenate((resultant\_string\_1,resultant\_string\_2),axis**=**1)

end\_string

model = LinearRegression()

model.fit(x,y)

Ridge(alpha=300)

**from** sklearn.model\_selection **import** train\_test\_split

​

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(

x,y,test\_size **=** 0.2,random\_state **=** 0)

# to find point of intercept and coefficient of the model

print(model.intercept\_,model.coef\_)

[35.47342376] [[0.82819723]]

predictionTestSet **=** model.predict(X\_test)

**from** sklearn.metrics **import** mean\_squared\_error

**from** sklearn.metrics **import** r2\_score

**import** numpy **as** np

​

y\_train\_predict **=** model.predict(X\_train)

rmse **=** (np.sqrt(mean\_squared\_error(y\_train, y\_train\_predict)))

r2 **=** r2\_score(y\_train, y\_train\_predict)

print("The model performance for training set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

​

The model performance for training set

--------------------------------------

RMSE is 76.92486402252042

R2 score is 0.6903635216011239

y\_test\_predict **=** model.predict(X\_test)

rmse **=** (np.sqrt(mean\_squared\_error(y\_test, y\_test\_predict)))

r2 **=** r2\_score(y\_test, y\_test\_predict)

​

print("The model performance for testing set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

The model performance for testing set

--------------------------------------

RMSE is 76.81764511153429

R2 score is 0.6915536114123604

model **=** LinearRegression()

model.fit(x,y)

LinearRegression()

model **=** LinearRegression().fit(x, y)

r\_sq **=** model.score(x, y)

print(f"coefficient of determination: {r\_sq}")

coefficient of determination: 0.690601967532237

print(f"intercept: {model.intercept\_}")

intercept: [35.47342376]

print(f"slope: {model.coef\_}")

slope: [[0.82819723]]

y\_pred **=** model.predict(X\_test)

plt.scatter(X\_test,y\_test, color **=** 'red')

plt.plot(X\_test, y\_pred, color **=** 'blue')

plt.title('Linear Regression')

plt.xlabel('Independent Variable')

plt.ylabel ('Dependent Variable')

plt.show()

Chart, scatter chart

Description automatically generated

**from** sklearn.metrics **import** mean\_squared\_error, r2\_score

print('Mean squared error: %.2f' **%** mean\_squared\_error(y\_test, y\_pred))

print('Coefficient of determination (R^2): %.2f' **%** r2\_score(y\_test, y\_pred))

Mean squared error: 5900.95

Coefficient of determination (R^2): 0.69

Q6.

The linear regression equation is y = 0.82819723x + 35.47342376.

The closer the value of R2 is to 1 will show how efficient the model is. The value of R2 is 0.69 which is quite close to 1 and we can infer that our model is quite efficient.

The value of r shows the relationship between the values of the x-axis and the values of the y-axis. If r is 0, it means the values are not related at all. If r is 1, it means the values are 100% related. In our model, the value of r is 0.831. This means that the values of x-axis and y-axis are quite related and have quite a strong relationship.

The Mean Squared Error measures how close the line is to a set of data points. In our model, the mean squared error is 5900.95 which means that the data is spread around widely. This is because there is a large amount of data and this could result in more error hence the mean squared error is large.

The slope of the line can be used to determine the rate of change and the slope of the model is 0.828 hence it means that the rate of change is not that fast.

Appendix

Q1)

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

insurance\_data.isnull().sum(axis **=** 0**/**1)

Q2)

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

updated\_insurance\_data **=** insurance\_data.dropna(axis**=**0, how **=**'any')

updated\_insurance\_data

updated\_insurance\_data.isnull().sum(axis **=** 0**/**1)

Q3)

(first)

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

second\_insurance\_data **=** insurance\_data.ffill(axis **=** 0**/**1)

second\_insurance\_data

(second)

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

insurance\_data.sort\_values(

by **=** ["Amount"], ascending**=False**)

(third)

import pandas as pd

import sklearn

import numpy as np

import matplotlib.pyplot as plt

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

insurance\_data.rename(columns **=** {'Claim\_ID' : 'A', 'Policy\_No' : 'B','Name' : 'C', 'Planned': 'D', 'Actual' : 'E', 'Created': 'F', 'Amount' : 'G', 'Paid':'H', 'Category' : 'I', 'Terms' : 'J', 'Region' : 'K', 'Type':'L'}, inplace **=** **True**)

insurance\_data

third\_insurance\_data **=** insurance\_data.drop(['A','B','C','D','E','F','H','J','K','L'], axis**=**1)

third\_insurance\_data

third\_insurance\_data.rename(columns **=** {'G': "Amount",'I': "Category"}, inplace **=** **True**)

third\_insurance\_data

Q5)

**import** pandas **as** pd

**import** sklearn

**from** sklearn.linear\_model **import** LinearRegression

**import** matplotlib.pyplot **as** plt

**from** scipy **import** stats

**import** numpy **as** np

insurance\_data **=** pd.read\_csv("ECA.csv")

insurance\_data

insurance\_data.rename(columns **=** {'Claim\_ID' : 'A', 'Policy\_No' : 'B','Name' : 'C', 'Planned': 'D', 'Actual' : 'E', 'Created': 'F', 'Amount' : 'G', 'Paid':'H', 'Category' : 'I', 'Terms' : 'J', 'Region' : 'K', 'Type':'L'}, inplace **=** **True**)

insurance\_data

fifth\_insurance\_data **=** insurance\_data.drop(['A','B','C','F','G','H','I','J','K','L'], axis**=**1)

fifth\_insurance\_data

fifth\_insurance\_data['D'] **=** pd.to\_datetime(fifth\_insurance\_data['D'])

fifth\_insurance\_data

fifth\_insurance\_data['E'] **=** pd.to\_datetime(fifth\_insurance\_data['E'])

fifth\_insurance\_data

updated\_fifth\_insurance\_data **=** fifth\_insurance\_data.dropna(axis**=**0, how **=**'any')

updated\_fifth\_insurance\_data

updated\_fifth\_insurance\_data['D\_delta'] **=** (updated\_fifth\_insurance\_data['D'] **-** updated\_fifth\_insurance\_data['D'].min())**/**np.timedelta64(1,'D')

updated\_fifth\_insurance\_data

updated\_fifth\_insurance\_data['E\_delta'] **=** (updated\_fifth\_insurance\_data['E'] **-** updated\_fifth\_insurance\_data['E'].min())**/**np.timedelta64(1,'D')

updated\_fifth\_insurance\_data

sixth\_insurance\_data **=** updated\_fifth\_insurance\_data.drop(['D','E'], axis**=**1)

sixth\_insurance\_data

sixth\_insurance\_data **=** sixth\_insurance\_data.astype({'D\_delta':'int'})

sixth\_insurance\_data

sixth\_insurance\_data **=** sixth\_insurance\_data.astype({'E\_delta':'int'})

sixth\_insurance\_data

**import** matplotlib.pyplot **as** plt

**from** scipy **import** stats

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_absolute\_error, mean\_squared\_error, r2\_score

x **=** sixth\_insurance\_data['D\_delta']

y **=** sixth\_insurance\_data['E\_delta']

**from** sklearn **import** preprocessing

**from** sklearn **import** linear\_model

array\_list\_1 **=** np.array([sixth\_insurance\_data['D\_delta']])

arr1 **=** array\_list\_1

resultant\_string\_1 **=** np.reshape(arr1, (22536, -1))

array\_list\_2 **=** np.array([sixth\_insurance\_data['E\_delta']])

arr2 **=** array\_list\_2

resultant\_string\_2 **=** np.reshape(arr2, (22536, **-**1))

x **=** resultant\_string\_1

y **=** resultant\_string\_2

end\_string **=** np.concatenate((resultant\_string\_1,resultant\_string\_2),axis**=**1)

end\_string

model **=** LinearRegression()

model.fit(x,y)

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(

x,y,test\_size **=** 0.2,random\_state **=** 0)

print(model.intercept\_,model.coef\_)

predictionTestSet **=** model.predict(X\_test)

**from** sklearn.metrics **import** mean\_squared\_error

**from** sklearn.metrics **import** r2\_score

**import** numpy **as** np

​

y\_train\_predict **=** model.predict(X\_train)

rmse **=** (np.sqrt(mean\_squared\_error(y\_train, y\_train\_predict)))

r2 **=** r2\_score(y\_train, y\_train\_predict)

print("The model performance for training set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

print("\n")

​

y\_test\_predict **=** model.predict(X\_test)

rmse **=** (np.sqrt(mean\_squared\_error(y\_test, y\_test\_predict)))

r2 **=** r2\_score(y\_test, y\_test\_predict)

​

print("The model performance for testing set")

print("--------------------------------------")

print('RMSE is {}'.format(rmse))

print('R2 score is {}'.format(r2))

model **=** LinearRegression()

model.fit(x,y)

model **=** LinearRegression().fit(x, y)

r\_sq **=** model.score(x, y)

print(f"coefficient of determination: {r\_sq}")

print(f"intercept: {model.intercept\_}")

print(f"slope: {model.coef\_}")

y\_pred **=** model.predict(X\_test)

plt.scatter(X\_test,y\_test, color **=** 'red')

plt.plot(X\_test, y\_pred, color **=** 'blue')

plt.title('Linear Regression')

plt.xlabel('Independent Variable')

plt.ylabel ('Dependent Variable')

plt.show()

**from** sklearn.metrics **import** mean\_squared\_error, r2\_score

print('Mean squared error: %.2f' **%** mean\_squared\_error(y\_test, y\_pred))

print('Coefficient of determination (R^2): %.2f' **%** r2\_score(y\_test, y\_pred))