- [Loading data: Drive, Sheets, and Google Cloud Storage]
- [Charts: visualizing data]
- [Getting started with BigQuery]

Machine Learning Crash Course

These are a few of the notebooks from Google's online Machine Learning course. See the full course website for more.

- Intro to Pandas DataFrame
- Linear regression with tf.keras using synthetic data

Using Accelerated Hardware

- TensorFlow with GPUs
- TensorFlow with TPUs

Featured examples

- NeMo Voice Swap: Use Nvidia's NeMo conversational AI Toolkit to swap a voice in an audio fragment with a computer generated one.
- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB movie reviews as either *positive* or *negative*.
- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
```

1. Download the dataset: Dataset
2. Load the dataset

```
data=pd.read_csv("Churn_Modelling.csv",encoding='ISO-8859-1')
```

```
data.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balan
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8

```
data.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000

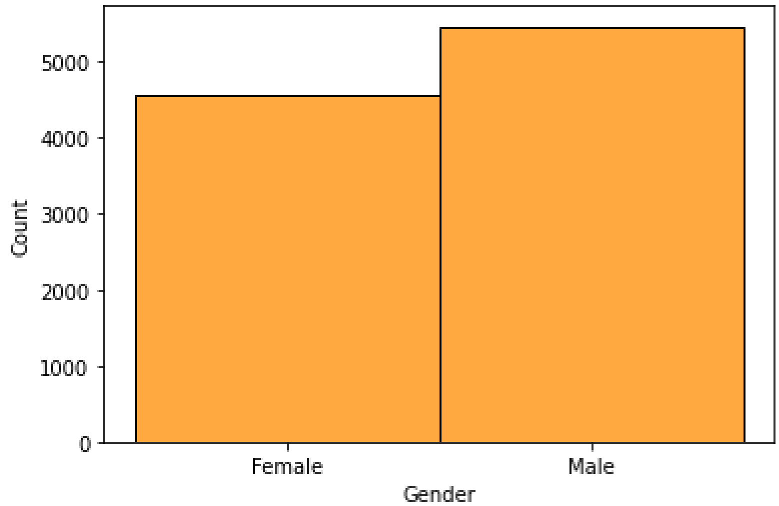
```
data.dtypes
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

3. Perform Below Visualizations Univariate Analysis ,Bi - Variate Analysis,Multi - Variate Analysis

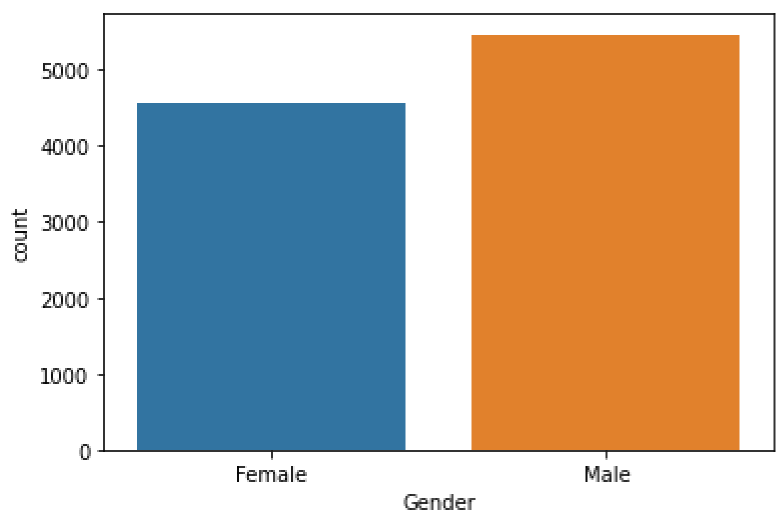
```
#univariate analysis "Histogram"
sns.histplot(data["Gender"],color='darkorange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f94851cab50>
```



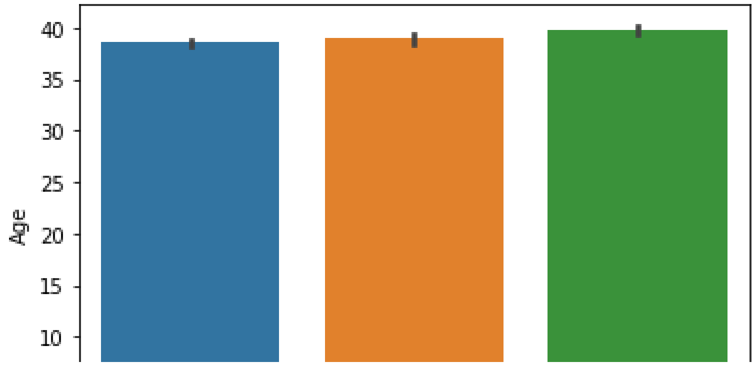

```
#univariate analysis "Countlot"
sns.countplot(data['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass tl
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f94851d1c50>
```



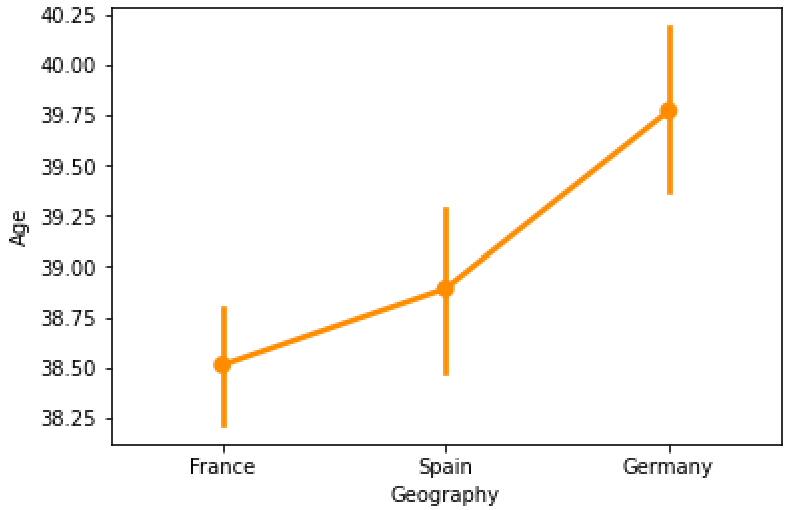

```
#bivariate analysis"Barplot"
sns.barplot(x='Geography',y='Age',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9484c19050>
```



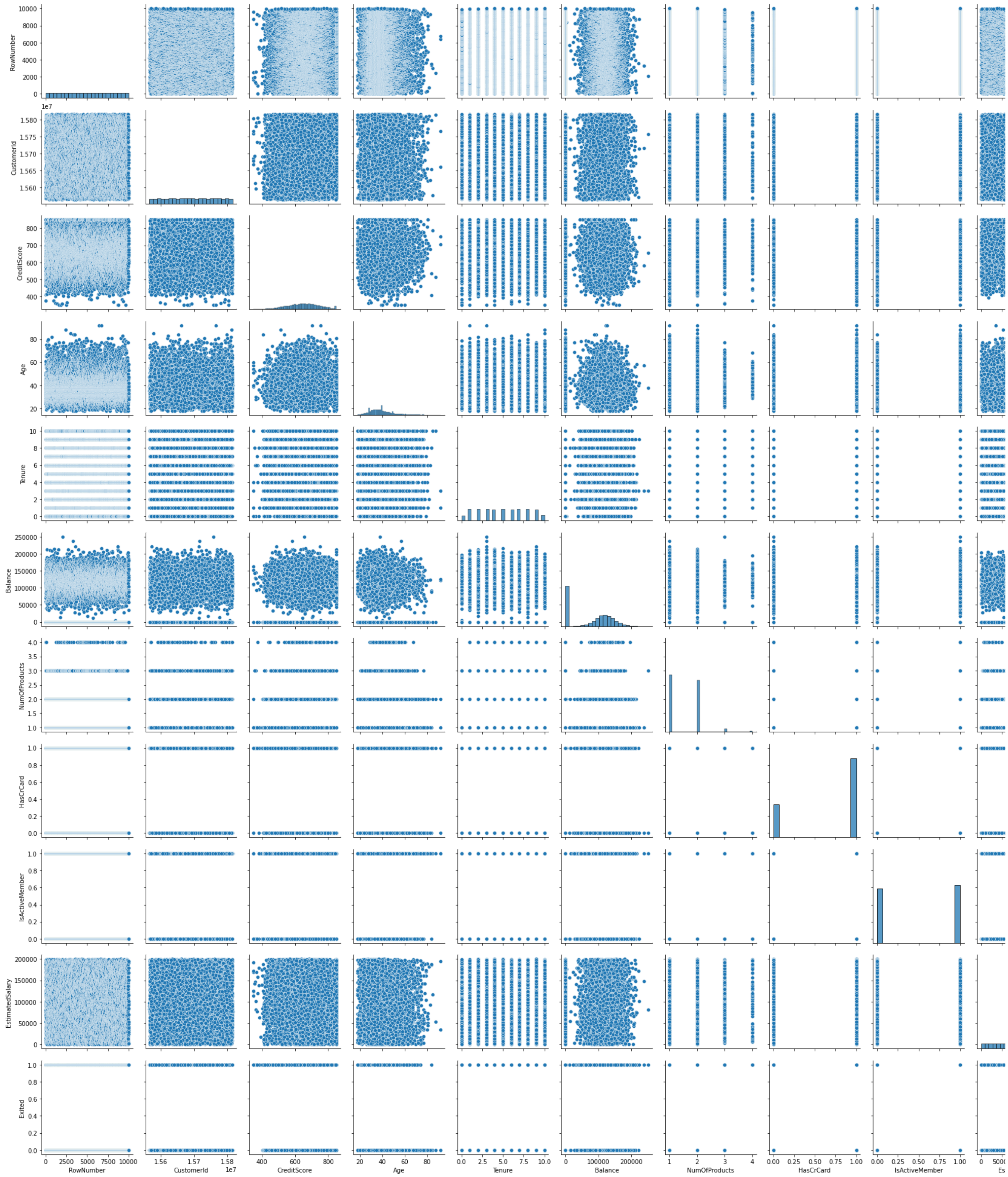

```
#bivariate analysis"Pointplot"
sns.pointplot(x='Geography',y='Age',data=data,color='darkorange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9484b8b2d0>
```




```
#Multivariate analysis"Pairplot"
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f9484b0b250>
```



4. ** Perform descriptive statistics on the dataset.**

```
# Descriptive statistics of the data set accessed.
data.describe().T
```

	count	mean	std	min	25%	5(
RowNumber	10000.0	5.000500e+03	2886.895680	1.00	2500.75	5.000500e+(
CustomerId	10000.0	1.569094e+07	71936.186123	15565701.00	15628528.25	1.569074e+(
CreditScore	10000.0	6.505288e+02	96.653299	350.00	584.00	6.520000e+(
Age	10000.0	3.892180e+01	10.487806	18.00	32.00	3.700000e+(
Tenure	10000.0	5.012800e+00	2.892174	0.00	3.00	5.000000e+(
Balance	10000.0	7.648589e+04	62397.405202	0.00	0.00	9.719854e+(
NumOfProducts	10000.0	1.530200e+00	0.581654	1.00	1.00	1.000000e+(
HasCrCard	10000.0	7.055000e-01	0.455840	0.00	0.00	1.000000e+(
IsActiveMember	10000.0	5.151000e-01	0.499797	0.00	0.00	1.000000e+(
EstimatedSalary	10000.0	1.000902e+05	57510.492818	11.58	51002.11	1.001939e+(
Exited	10000.0	2.037000e-01	0.402769	0.00	0.00	0.000000e+(

5. **Handle the Missing values.**

```
data.isnull().sum().sum()
```

```
0
```

This dataset does not contain any missing value.

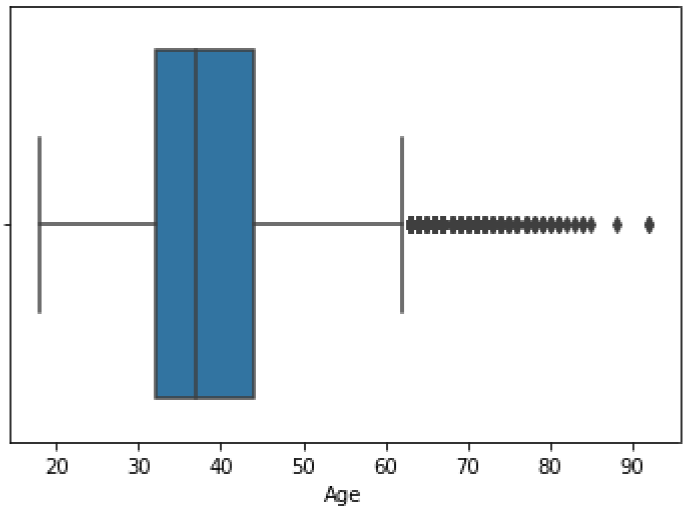
```
missing_values=data.isnull().sum()
missing_values[missing_values>0]/len(data)*100
```

```
Series([], dtype: float64)
```

6. **Find the outliers and replace the outliers**

```
sns.boxplot(data['Age'],data=data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f947fbcd650>
```



7. **Check for Categorical columns and perform encoding.**

```
print(data['Gender'].unique())
print(data['Age'].unique())
```

```
['Female' 'Male']
[42 41 39 43 44 50 29 27 31 24 34 25 35 45 58 32 38 46 36 33 40 51 61 49
 37 19 66 56 26 21 55 75 22 30 28 65 48 52 57 73 47 54 72 20 67 79 62 53
 80 59 68 23 60 70 63 64 18 82 69 74 71 76 77 88 85 84 78 81 92 83]
```

```
data['Gender'].value_counts()
data['Age'].value_counts()
```

```
37    478
38    477
35    474
36    456
34    447
     ...
92      2
82      1
88      1
85      1
83      1
Name: Age, Length: 70, dtype: int64
```

```
one_hot_encoded_data = pd.get_dummies(data, columns = ['Age', 'Gender'])
print(one_hot_encoded_data)
```

```
      RowNumber  CustomerId    Surname  CreditScore Geography  Tenure  \
0             1    15634602   Hargrave          619    France       2
1             2    15647311       Hill          608     Spain       1
2             3    15619304       Onio          502    France       8
3             4    15701354       Boni          699    France       1
4             5    15737888   Mitchell          850     Spain       2
...         ...         ...        ...          ...       ...     ...
9995       9996    15606229   Obijiaku          771    France       5
9996       9997    15569892  Johnstone          516    France      10
9997       9998    15584532        Liu          709    France       7
9998       9999    15682355  Sabbatini          772   Germany       3
9999      10000    15628319     Walker          792    France       4

        Balance  NumOfProducts  HasCrCard  IsActiveMember  ...  Age_80  \
0          0.00              1          1               1  ...       0
1      83807.86              1          0               1  ...       0
2     159660.80              3          1               0  ...       0
3          0.00              2          0               0  ...       0
4     125510.82              1          1               1  ...       0
...         ...            ...        ...             ...  ...     ...
9995       0.00              2          1               0  ...       0
9996   57369.61              1          1               1  ...       0
9997       0.00              1          0               1  ...       0
9998   75075.31              2          1               0  ...       0
9999  130142.79              1          1               0  ...       0

      Age_81  Age_82  Age_83  Age_84  Age_85  Age_88  Age_92  Gender_Female  \
0          0       0       0       0       0       0       0              1
1          0       0       0       0       0       0       0              1
2          0       0       0       0       0       0       0              1
3          0       0       0       0       0       0       0              1
4          0       0       0       0       0       0       0              1
...      ...     ...     ...     ...     ...     ...     ...            ...
9995       0       0       0       0       0       0       0              0
9996       0       0       0       0       0       0       0              0
9997       0       0       0       0       0       0       0              1
9998       0       0       0       0       0       0       0              0
9999       0       0       0       0       0       0       0              1

      Gender_Male
0               0
1               0
2               0
3               0
4               0
...           ...
9995            1
9996            1
9997            0
9998            1
```

```
9999            0

[10000 rows x 84 columns]
```

8. **Split the data into dependent and independent variables.**

```
from sklearn.datasets import load_iris


from sklearn import preprocessing
data = load_iris()

# separate the independent and dependent variables
X_data = data.data
target = data.target
print("Dependent variable")
print(X_data)
print("Independent variable")
print(target)
```

```
Dependent variable
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
```

```
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.  3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.  1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.  3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.  3.5 1.6 0.6]
[5.1 3.8 1.9 0.4]
[4.8 3.  1.4 0.3]
[5.1 3.8 1.6 0.2]
[4.6 3.2 1.4 0.2]
[5.3 3.7 1.5 0.2]
[5.  3.3 1.4 0.2]
[7.  3.2 4.7 1.4]
[6.4 3.2 4.5 1.5]
[6.9 3.1 4.9 1.5]
[5.5 2.3 4.  1.3]
[6.5 2.8 4.6 1.5]
[5.7 2.8 4.5 1.3]
```

9. Scale the independent variable**

```
# scale of independent variables
standard = preprocessing.scale(target)
print(standard)
```

```
[-1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487 -1.22474487
 -1.22474487 -1.22474487  0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
```

```
 1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
 1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
 1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
 1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
 1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487
 1.22474487  1.22474487  1.22474487  1.22474487  1.22474487  1.22474487]
```

10. **Split the data into training and testing**

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
# get the locations
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.05, random_state=0)
```