# Module 1: Data Science Fundamentals

## Module 1: Data Science Fundamentals

### Sprint 1: Data Wrangling and Storytelling

### Volcano Eruptions, once more!

# Background

For the last day of this sprint, we're analyzing volcano eruptions once more. Equipped with solid EDA knowledge, your objective is to look at the dataset with a creative spirit. We suggest tackling the task with the following framework:

1. Start by exploring the data, without any initial clear objective.
2. Raise hypotheses, answer them using the data.

---

# How to start?

So far, your objective has been to answer provided questions using the tools learned. Now, you have to ask questions yourself, and answer them using the data and EDA methods we've studied. How should I know what to ask? I've never seen this data (Okay, not exactly true, we've had a tiny exercise with it, but still).

As a reminder, start from EDA. Be creative! Examine each data source individually. Calculate some basic statistics, visualize the time data in relation to some numerical variable. Don't be afraid to group things and display them. Join data if you sense that enriching one datasource could give insights.

Having done these steps, you should be able to formulate a basic set of questions and start zooming in on the dataset - dissecting it further, along different, more specific dimensions, e.g. "Right, it's clear that rock A is the one errupted most. But maybe that's the case only for some specific set of big volcanoes?". And, down the rabbit hole you go!

## Concepts to explore

You should explore calculating basic data statistical parameters, performing EDA.

## Requirements

- [x] Describe the data with basic statistical parameters - mean, median, quantiles, etc.
- [x] Grouping the data and analyzing the groups - using pandas aggregate methods.
- [x] Work with features - handle missing data, use pandas date APIs.
- [x] Manipulate datasets - use joins.
- [x] Visualize the data - use line, scatter, histogram plots, density plots, regplots, etc.

The data is available here, you can use any of the datasets from the repository.

## Evaluation Criteria

- The code quality
- The quality of your raised hypotheses
- The quality of how methodologically you verified you hypotheses
- Adherence to the requirements

## Bonus challenges

- Can you enrich the data from sources other than the repository specified?
- Build a model to predict `major_rock_1` given `primary_volcano_type`.

---

# Data exploration on volcanoes

## Abstract

With data from The Smithsonian Institute this week's project is about exploring **volcanoes!** We'll dive into some interesting datasets about volcanoes, eruptions and tectonic plates and in the end we'll build a model to predict the `major_rock_1` given the `primary_volcano_type`. Let's get started!

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import folium
import datetime
import numpy as np

# Loading datasets into the dataframes
volcano = pd.read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesda
```

```
# Checking the size and shape of the dataframe to understand what we're working wit
print(f"The Volcano dataframe has a shape of {volcano.shape}, where the number {len
volcano.info()
```

```
The Volcano dataframe has a shape of (958, 26), where the number 958 represent

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 958 entries, 0 to 957
Data columns (total 26 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   volcano_number          958 non-null    int64
 1   volcano_name            958 non-null    object
 2   primary_volcano_type    958 non-null    object
 3   last_eruption_year      958 non-null    object
 4   country                 958 non-null    object
 5   region                  958 non-null    object
 6   subregion               958 non-null    object
 7   latitude                958 non-null    float64
 8   longitude               958 non-null    float64
 9   elevation               958 non-null    int64
 10  tectonic_settings       958 non-null    object
 11  evidence_category       958 non-null    object
 12  major_rock_1            958 non-null    object
 13  major_rock_2            958 non-null    object
 14  major_rock_3            958 non-null    object
 15  major_rock_4            958 non-null    object
 16  major_rock_5            958 non-null    object
 17  minor_rock_1            958 non-null    object
 18  minor_rock_2            958 non-null    object
 19  minor_rock_3            958 non-null    object
 20  minor_rock_4            958 non-null    object
```

```
 21  minor_rock_5              958 non-null    object
 22  population_within_5_km    958 non-null    int64
 23  population_within_10_km   958 non-null    int64
 24  population_within_30_km   958 non-null    int64
 25  population_within_100_km  958 non-null    int64
dtypes: float64(2), int64(6), object(18)
memory usage: 194.7+ KB
```

```
# Double check for null values
volcano.isnull().sum()
```

```
volcano_number              0
volcano_name                0
primary_volcano_type        0
last_eruption_year          0
country                     0
region                      0
subregion                   0
latitude                    0
longitude                   0
elevation                   0
tectonic_settings           0
evidence_category           0
major_rock_1                0
major_rock_2                0
major_rock_3                0
major_rock_4                0
major_rock_5                0
minor_rock_1                0
minor_rock_2                0
minor_rock_3                0
minor_rock_4                0
minor_rock_5                0
population_within_5_km      0
population_within_10_km     0
population_within_30_km     0
population_within_100_km    0
dtype: int64
```

```
# Check common statistics of the data
volcano.describe()
```

| | volcano_number | latitude | longitude | elevation | population_within_5_k |
|---|---|---|---|---|---|
| **count** | 958.000000 | 958.000000 | 958.000000 | 958.000000 | 9.580000e+0 |
| **mean** | 298585.325678 | 14.984680 | 23.537475 | 1867.027140 | 4.786046e+0 |
| **std** | 49792.657247 | 31.584983 | 109.852596 | 1401.545901 | 2.986690e+0 |
| **min** | 210010.000000 | -78.500000 | -179.970000 | -2500.000000 | 0.000000e+0 |
| **25%** | 263025.000000 | -5.401500 | -78.282750 | 881.000000 | 0.000000e+0 |
| **50%** | 300055.500000 | 14.514000 | 36.393500 | 1622.500000 | 2.950000e+0 |
| **75%** | 343088.000000 | 40.798250 | 131.045500 | 2548.250000 | 4.642000e+0 |
| **max** | 390829.000000 | 71.082000 | 179.580000 | 6879.000000 | 5.783287e+0 |

```
# Let's see what the dataframe looks like transposed
volcano.head(4).T
```

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **volcano_number** | 283001 | 355096 | 342080 | 213004 |
| **volcano_name** | Abu | Acamarachi | Acatenango | Acigol-Nevsehir |
| **primary_volcano_type** | Shield(s) | Stratovolcano | Stratovolcano(es) | Caldera |
| **last_eruption_year** | -6850 | Unknown | 1972 | -2080 |
| **country** | Japan | Chile | Guatemala | Turkey |
| **region** | Japan, Taiwan, Marianas | South America | México and Central America | Mediterranean and Western Asia |
| **subregion** | Honshu | Northern Chile, Bolivia and Argentina | Guatemala | Turkey |
| **latitude** | 34.5 | -23.292 | 14.501 | 38.537 |
| **longitude** | 131.6 | -67.618 | -90.876 | 34.621 |
| **elevation** | 641 | 6023 | 3976 | 1683 |
| **tectonic_settings** | Subduction zone / Continental crust (>25 km) | Subduction zone / Continental crust (>25 km) | Subduction zone / Continental crust (>25 km) | Intraplate / Continental crust (>25 km) |
| **evidence_category** | Eruption Dated | Evidence Credible | Eruption Observed | Eruption Dated |
| **major_rock_1** | Andesite / Basaltic Andesite | Dacite | Andesite / Basaltic Andesite | Rhyolite |
| **major_rock_2** | Basalt / Picro-Basalt | Andesite / Basaltic Andesite | Dacite | Dacite |
| **major_rock_3** | Dacite | | | Basalt / Picro-Basalt |
| **major_rock_4** | | | | Andesite / Basaltic Andesite |
| **major_rock_5** | | | | |
| **minor_rock_1** | | | Basalt / Picro-Basalt | |

There are a lot of volcano types, we can check the relation between the `primary_volcano_type` and the `elevation`. In addition, we can do something with the `latitude` and the `longitude` to plot the volcanoes on a map to see where they are.

```
volcano['primary_volcano_type'].value_counts()
```

```
    Stratovolcano          353
    Stratovolcano(es)      107
    Shield                  85
    Volcanic field          71
    Pyroclastic cone(s)     70
    Caldera                 65
    Complex                 46
    Shield(s)               33
    Submarine               27
    Lava dome(s)            26
    Fissure vent(s)         12
    Caldera(s)               9
    Compound                 9
    Maar(s)                  8
    Pyroclastic shield       7
    Tuff cone(s)             7
    Crater rows              5
    Subglacial               5
    Pyroclastic cone         4
    Lava dome                3
    Lava cone(s)             1
    Lava cone                1
    Stratovolcano?           1
    Lava cone(es)            1
    Tuff cone                1
    Complex(es)              1
    Name: primary_volcano_type, dtype: int64
```

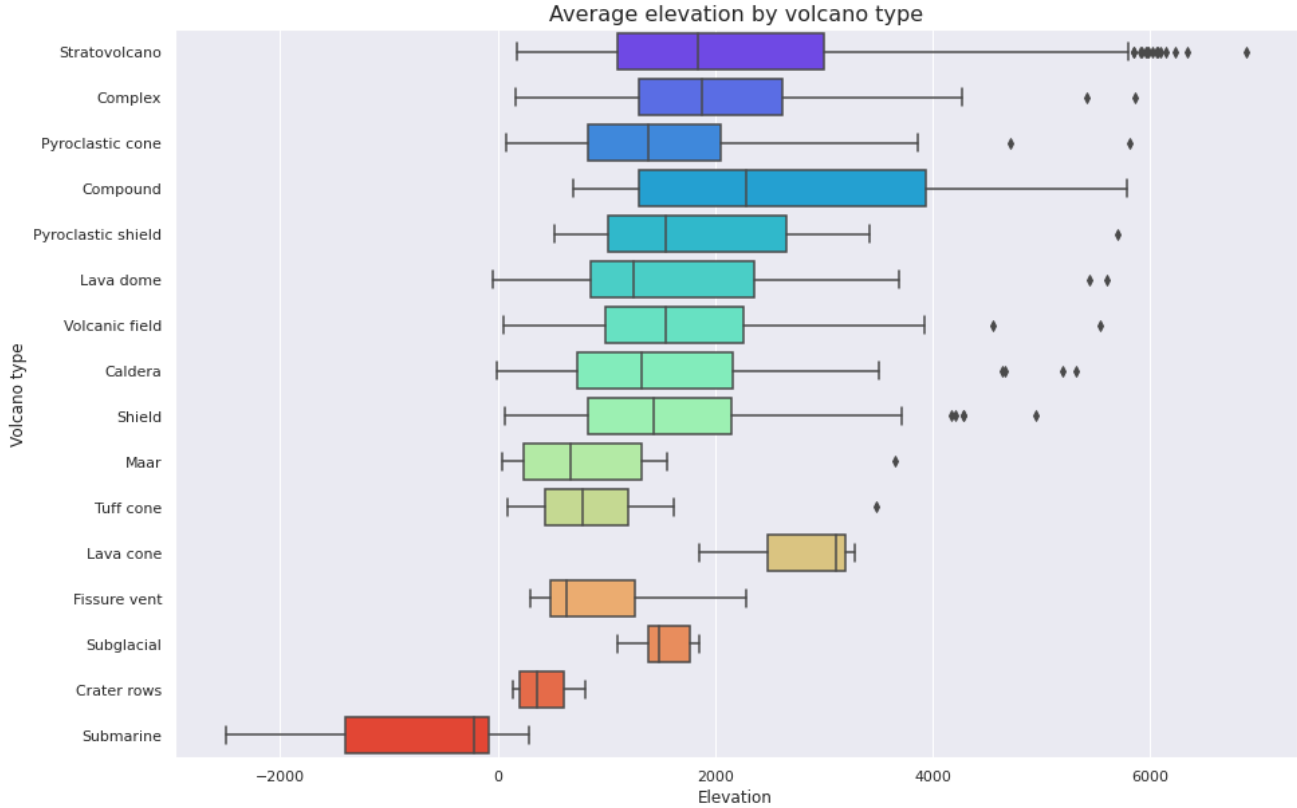
```
# group unneccesary extra characters like '(es)', '(s)' and '?'
volcano['primary_volcano_type'] = volcano['primary_volcano_type'].str.replace('\(es
volcano['primary_volcano_type'].value_counts()
```

```
    Stratovolcano         461
    Shield                118
    Caldera                74
    Pyroclastic cone       74
    Volcanic field         71
    Complex                47
    Lava dome              29
    Submarine              27
    Fissure vent           12
    Compound                9
    Tuff cone               8
    Maar                    8
    Pyroclastic shield      7
    Crater rows             5
    Subglacial              5
    Lava cone               3
    Name: primary_volcano_type, dtype: int64
```

```
# Group the elevation by primary_volcano_type
elevation = volcano[['elevation', 'primary_volcano_type']]
volcano_grouped_by_type = elevation.groupby('primary_volcano_type').mean().round(2)
volcano_grouped_by_type.nlargest(20, 'elevation')
```

| primary_volcano_type | elevation |
|---|---|
| Lava cone | 2751.33 |
| Compound | 2686.78 |
| Stratovolcano | 2229.05 |
| Pyroclastic shield | 2155.57 |
| Complex | 2047.17 |
| Lava dome | 1777.86 |
| Volcanic field | 1704.79 |
| Shield | 1634.25 |
| Pyroclastic cone | 1596.19 |
| Caldera | 1566.28 |
| Subglacial | 1518.00 |
| Tuff cone | 1065.62 |
| Maar | 1028.62 |
| Fissure vent | 948.17 |
| Crater rows | 422.60 |
| Submarine | -741.48 |

```
# Plot elevation by volcano type
volcano_sorted = volcano[['primary_volcano_type', 'elevation']].sort_values(by='ele
sns.set(style="darkgrid")
plt.figure(figsize=(15,10))

sns.boxplot(data=volcano_sorted, y='primary_volcano_type', x='elevation', palette='
plt.title("Average elevation by volcano type",
          horizontalalignment="center", fontsize=16)
plt.xlabel("Elevation")
plt.ylabel("Volcano type")

plt.show()
```




Nice! So we can see that the type of volcano does correlate with the elevation. A **submarine** volcano will most likely be **lower** than other types of volcanoes. Let's check some more data. Maybe the `region` or `country` can tell us something or the `tectonic_settings` shows interesting data.

```
# Check the regions
volcano.region.value_counts().head(10)
```

```
South America                     117
Japan, Taiwan, Marianas           102
Indonesia                          95
México and Central America         93
Africa and Red Sea                 79
Kamchatka and Mainland Asia        78
Canada and Western USA             60
Melanesia and Australia            45
Alaska                             38
Mediterranean and Western Asia     35
Name: region, dtype: int64
```

```
# Check the tectonic settings
volcano.tectonic_settings.value_counts()
```

```
Subduction zone / Continental crust (>25 km)        511
Intraplate / Continental crust (>25 km)             106
Subduction zone / Oceanic crust (< 15 km)            77
Rift zone / Continental crust (>25 km)               74
Rift zone / Oceanic crust (< 15 km)                  69
Subduction zone / Intermediate crust (15-25 km)      41
Subduction zone / Crustal thickness unknown          40
Rift zone / Intermediate crust (15-25 km)            21
Intraplate / Oceanic crust (< 15 km)                 14
```

```
Intraplate / Intermediate crust (15-25 km)           4
Unknown                                              1
Name: tectonic_settings, dtype: int64
```

```
# Let's plot all the volcanoes on a map!
volcano_map = folium.Map()

for i in range(0, volcano.shape[0]):
    volcano_col = volcano.iloc[i]
    folium.Marker([volcano_col['latitude'], volcano_col['longitude']], popup=volcan

volcano_map
```

Cool! Later on we'll come back to this map to plot the tectonic plate lines. In the meanwhile we'll look at the eruptions data. We can check the duration of eruptions with the `start_year`, `start_month`, `start_day`, `end_year`, `end_month` and `end_day` columns. In addition, the **V**olcano **E**xplosivity **I**ndex `vei` shows us how explosive a volcano is. Maybe there is a relation between the `duration` and the `vei`.

```
# Now, let's check eruptions data
# Load it in a dataframe
eruptions = pd.read_csv("https://raw.githubusercontent.com/rfordatascience/tidytues
eruptions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11178 entries, 0 to 11177
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   volcano_number         11178 non-null  int64
 1   volcano_name           11178 non-null  object
 2   eruption_number        11178 non-null  int64
 3   eruption_category      11178 non-null  object
 4   area_of_activity       4694 non-null   object
 5   vei                    8272 non-null   float64
 6   start_year             11177 non-null  float64
 7   start_month            10985 non-null  float64
 8   start_day              10982 non-null  float64
 9   evidence_method_dating  9898 non-null   object
 10  end_year               4332 non-null   float64
 11  end_month              4329 non-null   float64
 12  end_day                4326 non-null   float64
 13  latitude               11178 non-null  float64
 14  longitude              11178 non-null  float64
dtypes: float64(9), int64(2), object(4)
memory usage: 1.3+ MB
```

```
# Check null values & shape
print(f"The eruptions dataframe has a shape of {eruptions.shape}, where the number
eruptions.isnull().sum()
```

```
The eruptions dataframe has a shape of (11178, 15), where the number 11178 rep

volcano_number               0
volcano_name                 0
eruption_number              0
eruption_category            0
area_of_activity          6484
vei                       2906
start_year                   1
start_month                193
start_day                  196
evidence_method_dating    1280
end_year                  6846
end_month                 6849
end_day                   6852
latitude                     0
```

```
longitude              0
dtype: int64
```

```
eruptions.head(10)
```

| | volcano_number | volcano_name | eruption_number | eruption_category | area_of_ |
|---|---|---|---|---|---|
| 0 | 266030 | Soputan | 22354 | Confirmed Eruption | |
| 1 | 343100 | San Miguel | 22355 | Confirmed Eruption | |
| 2 | 233020 | Fournaise, Piton de la | 22343 | Confirmed Eruption | |
| 3 | 345020 | Rincon de la Vieja | 22346 | Confirmed Eruption | |
| 4 | 353010 | Fernandina | 22347 | Confirmed Eruption | |
| 5 | 273070 | Taal | 22344 | Confirmed Eruption | |
| 6 | 282050 | Kuchinoerabujima | 22345 | Confirmed Eruption | |
| 7 | 241040 | Whakaari/White Island | 22338 | Confirmed Eruption | 197 |
| 8 | 311060 | Semisopochnoi | 22341 | Confirmed Eruption | |
| 9 | 284096 | Nishinoshima | 22340 | Confirmed Eruption | |

```
# Check the minimum start year
print(eruptions['start_year'].min())

# Check eruption_categories
print(eruptions.eruption_category.unique())
print(eruptions.evidence_method_dating.unique())
```

```
-11345.0
['Confirmed Eruption' 'Uncertain Eruption' 'Discredited Eruption']
['Historical Observations' 'Seismicity' 'Hydrophonic' nan 'Uranium-series'
 'Magnetism' 'Radiocarbon (corrected)' 'Tephrochronology' 'Anthropology'
 'Lichenometry' 'Varve Count' 'Uncertain' 'Surface Exposure'
 'Radiocarbon (uncorrected)' 'Dendrochronology' 'Ice Core' 'Ar/Ar'
 'Hydration Rind' 'Fission track' 'Potassium-Argon' 'Thermoluminescence']
```

Meh, the eruption categories are not as thrilling as I thought. The `evidence_method_dating` however, is quite interesting. Maybe a certain method picks up more explosivity than the other. Keeping in mind that eruptions before the seismograph have to be calculated in another way, we might see that those methods show more explosivity since big, climate-changing eruptions happened a long time ago. We'll come back to this evidence later, first we'll look at the duration!

```
# Exclude 'trash' values
eruptions_year = eruptions.query("start_year > 1677 and start_month > 0 and start_d

# Start year, month, day to int
```

```
eruptions_year['start_year'] = eruptions_year['start_year'].astype(int)
eruptions_year['start_month'] = eruptions_year['start_month'].astype(int)
eruptions_year['start_day'] = eruptions_year['start_day'].astype(int)

# End year, month, day to int
eruptions_year['end_year'] = eruptions_year['end_year'].astype(int)
eruptions_year['end_month'] = eruptions_year['end_month'].astype(int)
eruptions_year['end_day'] = eruptions_year['end_day'].astype(int)

# Group by start year and get mean 'vei'
by_year_df = eruptions_year.groupby('start_year')
by_year_df['vei'].agg(['mean'])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
```

```
# Set full date
eruptions_year['full_date_start'] = eruptions_year['start_year'].astype(str) + '-'
eruptions_year['full_date_end'] = eruptions_year['end_year'].astype(str) + '-' + er

# Get the duration of eruptions
date = eruptions_year[['full_date_start']].apply(pd.to_datetime)
date['full_date_end'] = eruptions_year[['full_date_end']].apply(pd.to_datetime)
date['duration'] = date['full_date_end'] - date['full_date_start']
date['eruption_number'] = eruptions_year['eruption_number']
date
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
  This is separate from the ipykernel package so we can avoid doing imports ur
```

| | full_date_start | full_date_end | duration | eruption_number |
|---|---|---|---|---|
| **0** | 2020-03-23 | 2020-04-02 | 10 days | 22354 |
| **1** | 2020-02-22 | 2020-02-22 | 0 days | 22355 |
| **2** | 2020-02-10 | 2020-04-06 | 56 days | 22343 |
| **3** | 2020-01-31 | 2020-04-17 | 77 days | 22346 |
| **4** | 2020-01-12 | 2020-01-12 | 0 days | 22347 |
| **...** | ... | ... | ... | ... |
| **6901** | 1687-05-10 | 1687-05-11 | 1 days | 16609 |
| **6903** | 1687-03-26 | 1687-03-27 | 1 days | 10749 |
| **6906** | 1686-03-26 | 1686-03-27 | 1 days | 17575 |
| **6908** | 1685-10-03 | 1694-04-29 | 3130 days | 13336 |
| **6924** | 1682-08-12 | 1682-08-22 | 10 days | 13335 |

3413 rows × 4 columns

284 rows × 1 columns

```
date.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3413 entries, 0 to 6924
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   full_date_start  3413 non-null   datetime64[ns]
 1   full_date_end    3413 non-null   datetime64[ns]
 2   duration         3413 non-null   timedelta64[ns]
 3   eruption_number  3413 non-null   int64
dtypes: datetime64[ns](2), int64(1), timedelta64[ns](1)
memory usage: 293.3 KB
```

```
date.head()
```

| | full_date_start | full_date_end | duration | eruption_number |
|---|---|---|---|---|
| **0** | 2020-03-23 | 2020-04-02 | 10 days | 22354 |
| **1** | 2020-02-22 | 2020-02-22 | 0 days | 22355 |
| **2** | 2020-02-10 | 2020-04-06 | 56 days | 22343 |
| **3** | 2020-01-31 | 2020-04-17 | 77 days | 22346 |
| **4** | 2020-01-12 | 2020-01-12 | 0 days | 22347 |

```
# Let's add the VEI
date['vei'] = eruptions['vei']

# Drop null values
date_new = date.dropna(subset=['vei'])
date_new.head()
```

| | full_date_start | full_date_end | duration | eruption_number | vei |
|---|---|---|---|---|---|
| **7** | 2019-12-09 | 2019-12-09 | 0 days | 22338 | 2.0 |
| **9** | 2019-12-05 | 2020-04-17 | 134 days | 22340 | 1.0 |
| **13** | 2019-10-13 | 2019-10-22 | 9 days | 22334 | 1.0 |
| **25** | 2019-05-16 | 2019-10-07 | 144 days | 22320 | 2.0 |
| **32** | 2019-02-18 | 2019-07-28 | 160 days | 22306 | 2.0 |

```
# Check mean, max, min and quantiles for eruption duration in days
print(date_new.duration.mean())
print(date_new.duration.max())
print(date_new.duration.min())
print(date_new.duration.quantile([0.0, 0.25, 0.5, 0.75]))
```

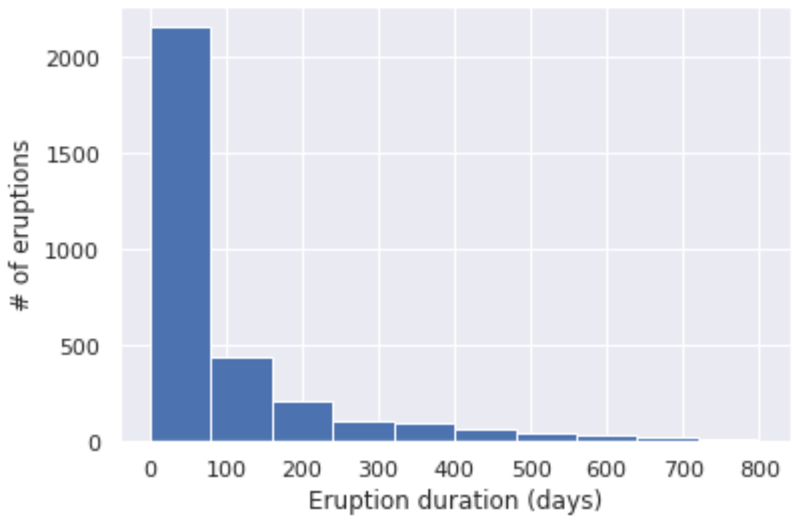
```
332 days 11:06:06.816952208
89774 days 00:00:00
0 days 00:00:00
```

```
0.00      0 days
0.25      3 days
0.50     35 days
0.75    153 days
Name: duration, dtype: timedelta64[ns]
```

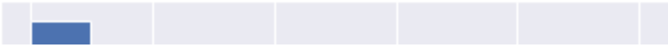
```
# Check mean, max, min and quantiles for eruption duration in days after excluding
date_normal=date_new[date_new.duration < datetime.timedelta(days=10_000)]
print(date_normal.duration.mean())
print(date_normal.duration.max())
print(date_normal.duration.min())
```

```
219 days 09:21:54.527333132
9381 days 00:00:00
0 days 00:00:00
```

```
# Plot the frequency of eruption duration in days
(date_normal.duration.astype('timedelta64[ns]') / pd.Timedelta(days=1)).hist(range=
plt.xlabel('Eruption duration (days)')
plt.ylabel('# of eruptions');
```




```
# Let's zoom in a bit more
(date_normal.duration.astype('timedelta64[ns]') / pd.Timedelta(days=1)).hist(range=
plt.xlabel('Eruption duration (days)')
plt.ylabel('# of eruptions');
```



Seems like most recent eruptions take less than 10 days. Now let's take a look at the `vei`.


```
# Group the data by year and get the sum vei per year
grouped = date_normal.groupby(pd.Grouper(key='full_date_start',freq='Y')).sum()

# Largest and smallest vei
print(grouped.nlargest(3, 'vei'))
print(grouped.nsmallest(3, 'vei'))
```

```
                 eruption_number   vei
full_date_start
2004-12-31                809465  89.0
2005-12-31                722980  77.0
2010-12-31                617802  76.0
                 eruption_number  vei
full_date_start
1683-12-31                     0  0.0
1684-12-31                     0  0.0
1688-12-31                     0  0.0
```

```
# Group the data by year and get the mean vei per year
grouped_day = date_normal.groupby(pd.Grouper(key='full_date_start',freq='Y')).mean(

# drop NaN
grouped_day.dropna(subset=['vei'])

# Largest and smallest vei
print(grouped_day.nlargest(3, 'vei'))
print(grouped_day.nsmallest(3, 'vei'))
```

```
                 eruption_number  vei
full_date_start
1721-12-31               12673.0  5.0
1739-12-31               18612.0  5.0
1693-12-31               12743.0  4.0
                 eruption_number  vei
full_date_start
1757-12-31               12992.0  0.0
1702-12-31               13699.0  1.0
1726-12-31               12874.0  1.0
```

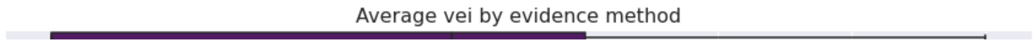
2004 has the highest `vei` and could be named *the most explosive year*, while the mean `vei` per year is the highest in 1721. This could be due to the fact that 2004 had many less-explosive (smaller) eruptions. To come back to the `evidence_method_dating`, we'll see which methods pick up the highest explosivity and vice versa.

```
# create new evidence dataframe
evidence = eruptions

# drop null values from the evidence_method_dating column
evidence.dropna(subset=['evidence method dating'], inplace=True)
```

```
evidence.evidence_method_dating.unique()
```

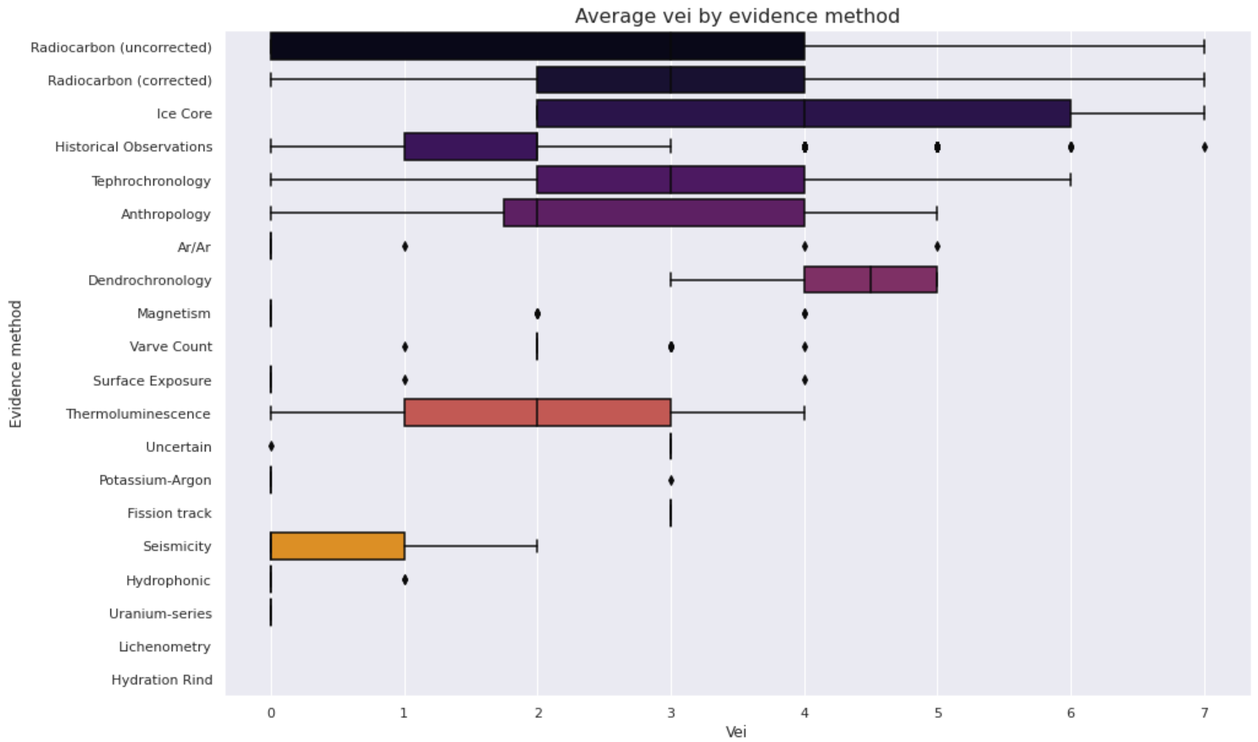
```
array(['Historical Observations', 'Seismicity', 'Hydrophonic',
       'Uranium-series', 'Magnetism', 'Radiocarbon (corrected)',
       'Tephrochronology', 'Anthropology', 'Lichenometry', 'Varve Count',
       'Uncertain', 'Surface Exposure', 'Radiocarbon (uncorrected)',
       'Dendrochronology', 'Ice Core', 'Ar/Ar', 'Hydration Rind',
       'Fission track', 'Potassium-Argon', 'Thermoluminescence'],
      dtype=object)
```

```
# Plot the evidence method by vei while checking the eruption category
evidence_sorted = evidence[['evidence_method_dating', 'vei', 'eruption_category']].
sns.set(style="darkgrid")
plt.figure(figsize=(15,10))

sns.boxplot(data=evidence_sorted, y='evidence_method_dating', x='vei', palette='inf
plt.title("Average vei by evidence method",
          horizontalalignment="center", fontsize=16)
plt.xlabel("Vei")
plt.ylabel("Evidence method")

plt.show()
```




Well.. all the above data comes from confirmed eruptions (duh). Better to leave `eruption_category` out to get a clearer look at the data!


```
# Plot the evidence method by vei
evidence_sorted2 = evidence[['evidence_method_dating', 'vei']].sort_values(by='vei'
sns.set(style="darkgrid")
plt.figure(figsize=(15,10))

sns.boxplot(data=evidence_sorted2, y='evidence_method_dating', x='vei', palette='in
plt.title("Average vei by evidence method",
          horizontalalignment="center", fontsize=16)
plt.xlabel("Vei")
plt.ylabel("Evidence method")

plt.show()
```



That's better! In line of what was expected, seismicity has a low `vei` because it is being used to keep track of recent eruptions and the most explosive eruptions can be found in with looking at the ice core. Finally, I would be interested to see which `primary_volcano_type` has the highest `vei`. In order to do that we must merge the two datasets used before.

```
# Load the datasets
volcano_for_merge = pd.read_csv("https://raw.githubusercontent.com/rfordatascience/
eruptions_for_merge = pd.read_csv("https://raw.githubusercontent.com/rfordatascienc

# Merge! (default is inner, which we want)
merged = pd.merge(volcano_for_merge, eruptions_for_merge, on='volcano_number')
merged.head()
```

| | volcano_number | volcano_name_x | primary_volcano_type | last_eruption_year | |
|---|---|---|---|---|---|
| **0** | 283001 | Abu | Shield(s) | -6850 | |
| **1** | 342080 | Acatenango | Stratovolcano(es) | 1972 | G |
| **2** | 342080 | Acatenango | Stratovolcano(es) | 1972 | G |
| **3** | 342080 | Acatenango | Stratovolcano(es) | 1972 | G |
| **4** | 342080 | Acatenango | Stratovolcano(es) | 1972 | G |

```
# check for null values
merged.isnull().sum()
```

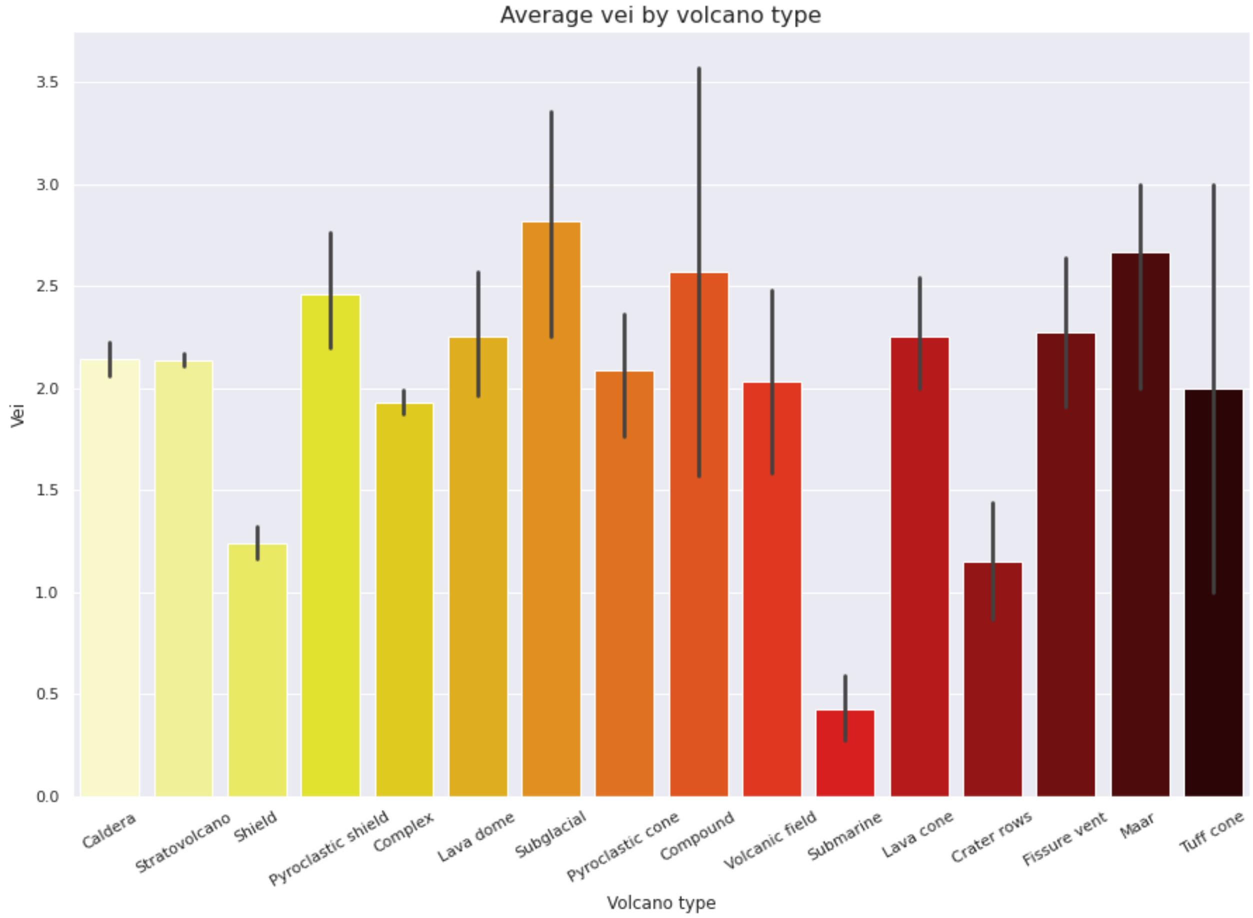
```
volcano_number              0
volcano_name_x              0
primary_volcano_type        0
last_eruption_year          0
country                     0
region                      0
subregion                   0
latitude_x                  0
longitude_x                 0
elevation                   0
tectonic_settings           0
evidence_category           0
major_rock_1                0
major_rock_2                0
```

```
major_rock_3                  0
major_rock_4                  0
major_rock_5                  0
minor_rock_1                  0
minor_rock_2                  0
minor_rock_3                  0
minor_rock_4                  0
minor_rock_5                  0
population_within_5_km        0
population_within_10_km       0
population_within_30_km       0
population_within_100_km      0
volcano_name_y                0
eruption_number               0
eruption_category             0
area_of_activity           5212
vei                        2399
start_year                    1
start_month                 171
start_day                   174
evidence_method_dating     1007
end_year                   5865
end_month                  5868
end_day                    5871
latitude_y                    0
longitude_y                   0
dtype: int64
```

```
# drop null values for vei
merged.dropna(subset=['vei'], inplace=True)
print(merged.vei.isnull().sum())

# group unneccesary extra characters like '(es)', '(s)' and '?'
merged['primary_volcano_type'] = merged['primary_volcano_type'].str.replace('\(es\)
```

```
0
```

```
# Plot the evidence method by vei
merged_sorted = merged[['primary_volcano_type', 'vei']].sort_values(by='vei', ascen
sns.set(style="darkgrid")
plt.figure(figsize=(15,10))

sns.barplot(data=merged_sorted, x='primary_volcano_type', y='vei', palette='hot_r')
plt.title("Average vei by volcano type",
          horizontalalignment="center", fontsize=16)
plt.ylabel("Vei")
plt.xlabel("Volcano type")
plt.xticks(rotation=30)


plt.show()
```




Voila! As we can see, Subglacial volcanoes tend to be the most explosive. This explains why the `evidence_method_type` *Ice core* has the highest `vei`.

# Bonus Challenges

## Challenge 1 - Enrich dataset

We will plug in the tectonic plate lines in the volcano map from before

```
# Load the tectonic plate data
tectonic = pd.read_csv("https://raw.githubusercontent.com/TuringCollegeSubmissions/
tectonic.shape
```

```
(12321, 3)
```

```
tectonic.head(5)
```

| | plate | lat | lon |
|---|---|---|---|
| **0** | am | 30.754 | 132.824 |
| **1** | am | 30.970 | 132.965 |
| **2** | am | 31.216 | 133.197 |
| **3** | am | 31.515 | 133.500 |

```
# Check for null values
tectonic.isnull().sum()
```

```
plate    0
lat      0
lon      0
dtype: int64
```

```
# Create the tectonic_plate map
plate_map = folium.Map()

plates = list(tectonic['plate'].unique())

for plate in plates:
    plate_vals = tectonic[tectonic['plate'] == plate]
    lats = plate_vals['lat'].values
    lons = plate_vals['lon'].values
    points = list(zip(lats, lons))
    indexes = [None] + [i + 1 for i, x in enumerate(points) if i < len(points) - 1
    for i in range(len(indexes) - 1):
        folium.vector_layers.PolyLine(points[indexes[i]:indexes[i+1]], popup=plate,

plate_map
```

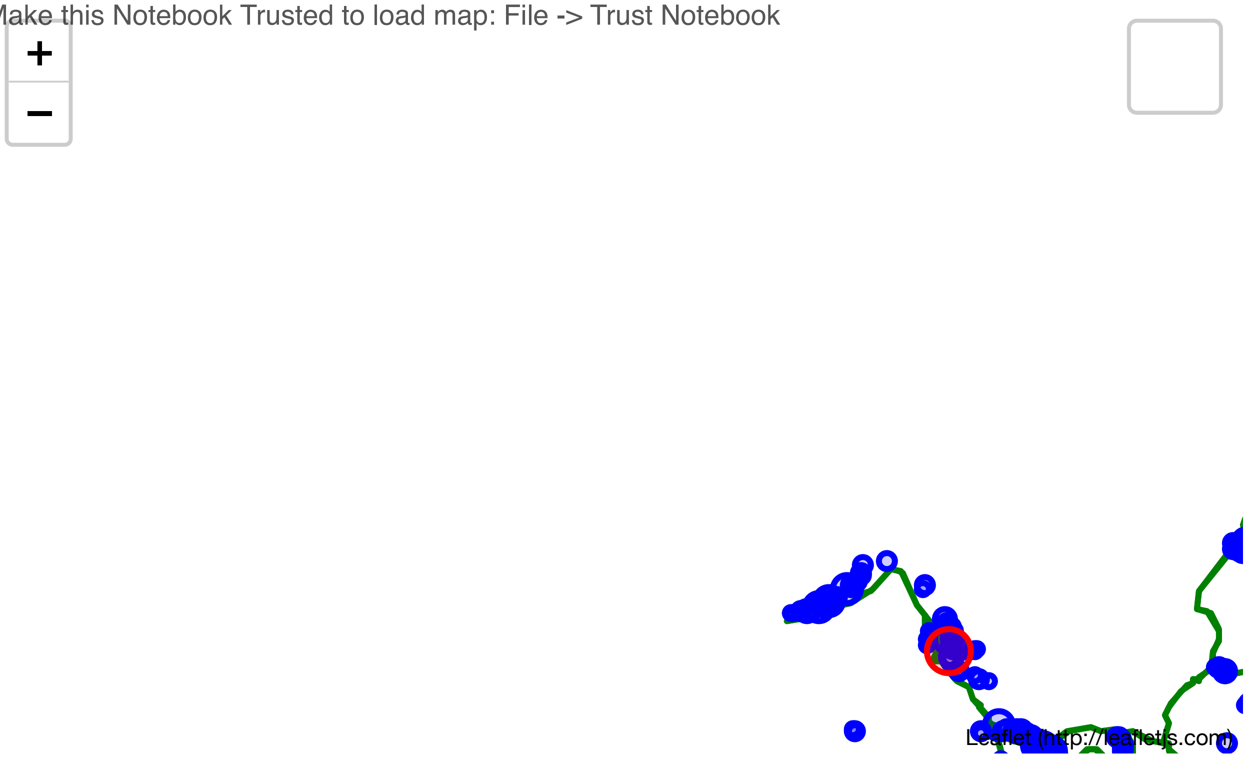
Make this Notebook Trusted to load map: File -> Trust Notebook


```
# Get max VEI for each volcano
volcano_max_vei = eruptions.groupby(['volcano_number'])['vei'].max().reset_index()

# Merge into data_volcano dataframe
data_volcano = pd.merge(volcano, volcano_max_vei, on='volcano_number')

# Check null values
data_volcano.isnull().sum()
```

```
volcano_number              0
volcano_name                0
primary_volcano_type        0
last_eruption_year          0
country                     0
region                      0
subregion                   0
latitude                    0
longitude                   0
elevation                   0
tectonic_settings           0
evidence_category           0
major_rock_1                0
major_rock_2                0
major_rock_3                0
major_rock_4                0
major_rock_5                0
minor_rock_1                0
minor_rock_2                0
minor_rock_3                0
minor_rock_4                0
minor_rock_5                0
population_within_5_km      0
population_within_10_km     0
population_within_30_km     0
population_within_100_km    0
vei                       152
dtype: int64
```

```
def vei_radius(vei):
    return 2 ** (int(vei) - 4) + 3 if not np.isnan(vei) else 1

volcano_with_vei = data_volcano.dropna(subset=['vei'])

# Create the map
complete_map = folium.Map()

# Add tectonic plates to map
plate_layer = folium.FeatureGroup(name='Tectonic Plates')
plates = list(tectonic['plate'].unique())
for plate in plates:
    plate_vals = tectonic[tectonic['plate'] == plate]
    lats = plate_vals['lat'].values
    lons = plate_vals['lon'].values
    points = list(zip(lats, lons))
    indexes = [None] + [i + 1 for i, x in enumerate(points) if i < len(points) - 1
    for i in range(len(indexes) - 1):
        folium.vector_layers.PolyLine(points[indexes[i]:indexes[i+1]], popup=plate,
plate_layer.add_to(complete_map)

# Create layers
layers = []
for i in range(8):
    layers.append(folium.FeatureGroup(name='VEI: '+str(i)))
layers.append(folium.FeatureGroup(name='VEI: NaN'))

# Add each volcano to the correct layer
for i in range(0, volcano_with_vei.shape[0]):
    volcano = volcano_with_vei.iloc[i]
    # Create marker
    marker = folium.CircleMarker([volcano['latitude'],
                                  volcano['longitude']],
                                  popup=volcano['volcano_name'] + ', VEI: ' + str(v
                                  radius=vei_radius(volcano['vei']),
                                  color='red' if not np.isnan(volcano['vei']) and i
                                  fill=True)
    # Add to correct layer
    if np.isnan(volcano['vei']):
        marker.add_to(layers[8])
    else:
        marker.add_to(layers[int(volcano['vei'])])

# Add layers to map
for layer in layers:
    layer.add_to(complete_map)

# Add layer control
folium.LayerControl().add_to(complete_map)

complete_map
```

Make this Notebook Trusted to load map: File -> Trust Notebook




## Challenge 2 - Make predictions

We will predict `major_rock_1` given the `primary_volcano_type` from the volcano dataset.

```
# Load the volcano dataset
volcano_model = pd.read_csv("https://raw.githubusercontent.com/rfordatascience/tidy

# Clean primary_volcano_types
volcano_model['primary_volcano_type'] = volcano_model['primary_volcano_type'].str.r
print(volcano_model["primary_volcano_type"].value_counts())

# Set primary_volcano_type as category
volcano_model["primary_volcano_type"] = volcano_model["primary_volcano_type"].astyp
```

```
volcano_model.dtypes
```

```
Stratovolcano         461
Shield                118
Caldera                74
Pyroclastic cone       74
Volcanic field         71
Complex                47
Lava dome              29
Submarine              27
Fissure vent           12
Compound                9
Tuff cone               8
Maar                    8
Pyroclastic shield      7
Crater rows             5
Subglacial              5
Lava cone               3
Name: primary_volcano_type, dtype: int64
volcano_number                 int64
volcano_name                  object
primary_volcano_type        category
last_eruption_year            object
country                       object
region                        object
subregion                     object
latitude                     float64
longitude                    float64
elevation                      int64
tectonic_settings             object
evidence_category             object
major_rock_1                  object
major_rock_2                  object
major_rock_3                  object
major_rock_4                  object
major_rock_5                  object
minor_rock_1                  object
minor_rock_2                  object
minor_rock_3                  object
minor_rock_4                  object
minor_rock_5                  object
population_within_5_km         int64
population_within_10_km        int64
population_within_30_km        int64
population_within_100_km       int64
dtype: object
```

```
# Assign encoded variables to new column primary_volcano_type_cat
volcano_model["primary_volcano_type_cat"] = volcano_model["primary_volcano_type"].c
volcano_model.head()
```

| | volcano_number | volcano_name | primary_volcano_type | last_eruption_year | co |
|---|---|---|---|---|---|
| 0 | 283001 | Abu | Shield | -6850 | |
| 1 | 355096 | Acamarachi | Stratovolcano | Unknown | |
| 2 | 342080 | Acatenango | Stratovolcano | 1972 | Gua |

```
# Set input and output set
X = volcano_model[['primary_volcano_type_cat']]
y = volcano_model[['major_rock_1']]

print(X["primary_volcano_type_cat"].value_counts())
print(X.dtypes)

# Check for null values
print(X.isnull().sum())
print(y.isnull().sum())
```

```
11    461
10    118
8      74
0      74
15     71
1      47
6      29
13     27
4      12
2       9
14      8
7       8
9       7
12      5
3       5
5       3
Name: primary_volcano_type_cat, dtype: int64
primary_volcano_type_cat    int8
dtype: object
primary_volcano_type_cat    0
dtype: int64
major_rock_1    0
dtype: int64
```

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()

# Fit the model
model.fit(X, y)

# test with 'Shield' and 'Stratovolcano'
```

```
predictions = model.predict([ [10],[11] ])
print(predictions)
```

```
['Basalt / Picro-Basalt' 'Andesite / Basaltic Andesite']
```

```
# With an accuracy score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)

score = accuracy_score(y_test, predictions)
print(score)
```

```
0.609375
```