Volcano Eruptions

Prediction Model

Students:

Ram Michaeli, 318353364
Gal Israeli, 208416750
Yulia Kuderko, 321117863

RESEARCH QUESTION

Can we predict how powerful will the next volcano eruption be?

What pattern can we observe and what conclusions can we learn from them?

Phases

Data acquisition

Data cleaning

EDA

Machine Learning

Predict volcano type

Predict So2 emission of volcano

Prediction of next eruption strength

Conclusions


DATA
ACQUISITION

CRAWLING PROCESS

Crawling

1) First, we used selenium in order to be able to click on the “submit eruptions” button that would lead us to the main url, which we needed to collect our data from.
2) We created lists for the features that eventually will be represented by columns in the final data frame.
3) Then, we used a loop that went through the pages using selenium library. Selenium allowed us to click the "next" button after every time we finished collecting the data from the current page.
4) We were able to extract all the relevant attributes and instances in each specific page using beautifulSoup library.
5) Each attribute was added to the lists we created in the beginning of the code.
6) Eventually, the data was exported to a csv file. This file is required in order to start analyzing our data in Jupyter notebook.

EXPORTED DATA FRAME

	volcanoName	eruptionCertainty	startDate	maxVEI	activityArea	evidenceMethod	stopDate	latitude	longitude	volcanoHeight	country	primaryVolcan
0	Ambae	Confirmed Eruption	2021 Dec 5	--	NaN	Historical Observations	2021 Dec 9	-15.389	167.835	1496 m	Vanuatu	
1	Pinatubo	Confirmed Eruption	2021 Nov 30	--	NaN	Historical Observations	2021 Nov 30	15.130	120.350	1486 m	Philippines	Stratov
2	Iliwerung	Confirmed Eruption	2021 Nov 28	--	NaN	Historical Observations	2021 Nov 29	-8.532	123.573	583 m	Indonesia	Cc
3	Taal	Confirmed Eruption	2021 Nov 15	--	NaN	Historical Observations	2021 Nov 22	14.002	120.993	311 m	Philippines	C
4	Turrialba	Confirmed Eruption	2021 Nov 3	--	NaN	Historical Observations	2021 Nov 23	10.025	-83.767	3340 m	Costa Rica	Stratov
...	...	...	...	...	...	...	...	...	...	...	...	
11217	Yojoa, Lago	Confirmed Eruption	1073 BCE	--	NaN	Radiocarbon (corrected)	Unknown	14.964	-87.983	1060 m	Honduras	Volcan
11218	Quetrupillan	Confirmed Eruption	1345 BCE	3	Quet2	Radiocarbon (corrected)	Unknown	-39.496	-71.722	2360 m	Chile	Stratov
11219	Saunders	Confirmed Eruption		0	NaN	Unknown	Unknown	-57.800	-26.483	843 m	United Kingdom	Stratov
11220	Fournaise, Piton de la	Confirmed Eruption		--	NaN	Unknown	Unknown	-21.244	55.708	2632 m	France	
11221	Karangetang	Confirmed Eruption		--	NaN	Unknown	Unknown	2.781	125.407	1797 m	Indonesia	Stratov

11222 rows × 16 columns

Crawling resources

Website that was used for the acquisition process	*Smithsonian Institute - National Museum of Natural History*
Packages that were used for crawling	*BeautifulSoup, Selenium, pandas, requests* *Minor packages: os, re, sys, time, warnings*
The attributes (columns)	*Volcano name, Eruption certainty, Start date, Max VEI, Evidence method, Stop date, Latitude, Longitude, Volcano height, Country, Primary volcano type, Population 5km, Population 10km, Population 30km, Population 100km*
Amount of data that was collected (Raw data volume)	*179,552 (before cleaning) ; 148,904 (after cleaning)*


DATA
CLEANING

HANDLING MISSING DATA

Filling data ('Max VEI' and 'Evidence Method')

- Filling Max VEI with mean values. We took each volcano that had several eruptions, calculated their Max VEI averages and added them to the missing cells. Those who had no Max VEI history at all, were dropped completely.

```
maxVEI_dict = dict()
for i,j in df.iterrows():
    if (pd.isna(j['maxVEI'])):
        continue
    else:
        if j['volcanoName'] in maxVEI_dict.keys():
            maxVEI_dict[j['volcanoName']].append(j['maxVEI'])
        else:
            maxVEI_dict[j['volcanoName']]=list([j['maxVEI']])
def cal_average(num):
    sum_num = 0
    for t in num:
        sum_num = sum_num + int(t)
    avg = sum_num / len(num)
    return avg
all_avgs=dict()
for key in maxVEI_dict:
    all_avgs[key]=(int(cal_average(maxVEI_dict[key])))
for i, j in df.iterrows():
    if (pd.isna(j['maxVEI']) and j['volcanoName'] in all_avgs):
        df.at[i,'maxVEI']=all_avgs[j['volcanoName']]
```

- Filling 'Evidence Method' with upper neighbor value. We filled each missing cell with its neighbor's value using the method: "ffill".

Filling Evidence Method with upper neighbour value

```
df['evidenceMethod'].fillna(method='ffill',inplace=True)

df['evidenceMethod'].isnull().sum()
```

```
0
```

Dropping null values

- "Activity Area" and "Stop Date" columns ware erased since more than a half of the data was empty. Therefore, these columns are irrelevant.

Activity Area

```
rowsWithArea = df.activityArea.value_counts().to_frame().sum()
totalRows = df.shape[0]

print(str((1- round((rowsWithArea/totalRows)[0],2)))+"% volcanoes don't have activityArea value!")
# Over 50% of the data don't have "activityArea" so we will remove this column

df.drop('activityArea', axis=1, inplace=True)

'activityArea' not in df.columns
```

```
0.54% volcanoes don't have activityArea value!
True
```

- After taking care of all the cells, we dropped each row that contained missing values using the "dropna" method.

```
# We can see that only 3 rows remain with missing values so now we will drop them
df.dropna(inplace=True)
df.isnull().sum()
```

```
volcanoName          0
eruptionCertainty    0
startDate            0
maxVEI               0
evidenceMethod       0
latitude             0
longitude            0
volcanoHeight        0
country              0
primaryVolcanoType   0
population5km        0
population10km       0
population30km       0
population100km      0
dtype: int64
```

CONVERTING TYPES

Numeric Type

- In order to change the 'population' columns to numeric values, we had to replace the coma character to an empty character and then we could switch to the desired type.
- In addition, we had to replace the "m" letter to an empty character so there wouldn't be a mix up between numeric characters and strings.

After both of these actions, we could easily implement the appropriate type using the "to_numeric" function.

```
cols = ['maxVEI', 'latitude', 'longitude', 'population5km',
        'population10km', 'population30km','population100km', 'volcanoHeight']
problems = ['population5km', 'population10km', 'population30km', 'population100km']
df[problems] = df[problems].replace(',','',regex=True)
df['volcanoHeight'] = df['volcanoHeight'].replace('m','',regex=True)
df[cols] = df[cols].apply(pd.to_numeric)
```

Categorical Type

As we can see, 'Eruption Certainty' has only three possible values: confirmed eruption, uncertain eruption and discredited eruption. Therefore, we changed their string values into numeric categorial so that our analyzation in the ML section would be more convenient.

```
df['eruptionCertainty'].unique()
```

```
array(['Confirmed Eruption', 'Uncertain Eruption', 'Discredited Eruption'],
      dtype=object)
```

```
replace_map = {'Confirmed Eruption': 1, 'Uncertain Eruption': 2, 'Discredited Eruption': 3}
df.replace(replace_map, inplace=True)
df.head()
```

OUTLIERS

- Overall, we had almost all our data in its supposed range except for the "volcano height" column.
- According to google, the deepest volcano in the world is -4200 meters, so we added the "drop" function in order to eliminate all the volcano heights that exceed this limitation.

```
# We know that the highest volcano in the world is: "Ojos del Salado" and its height is ~ 6800
# We know that the deepest volcano in the world is: "Pico Fracture Zone" and its height is ~ -4200
print(df.volcanoHeight.max())
print(df.volcanoHeight.min())
```

```
6879
-5700
```

```
df.drop(df.index[df['volcanoHeight'] < (-4200)], inplace=True)
print(df.volcanoHeight.min())
```

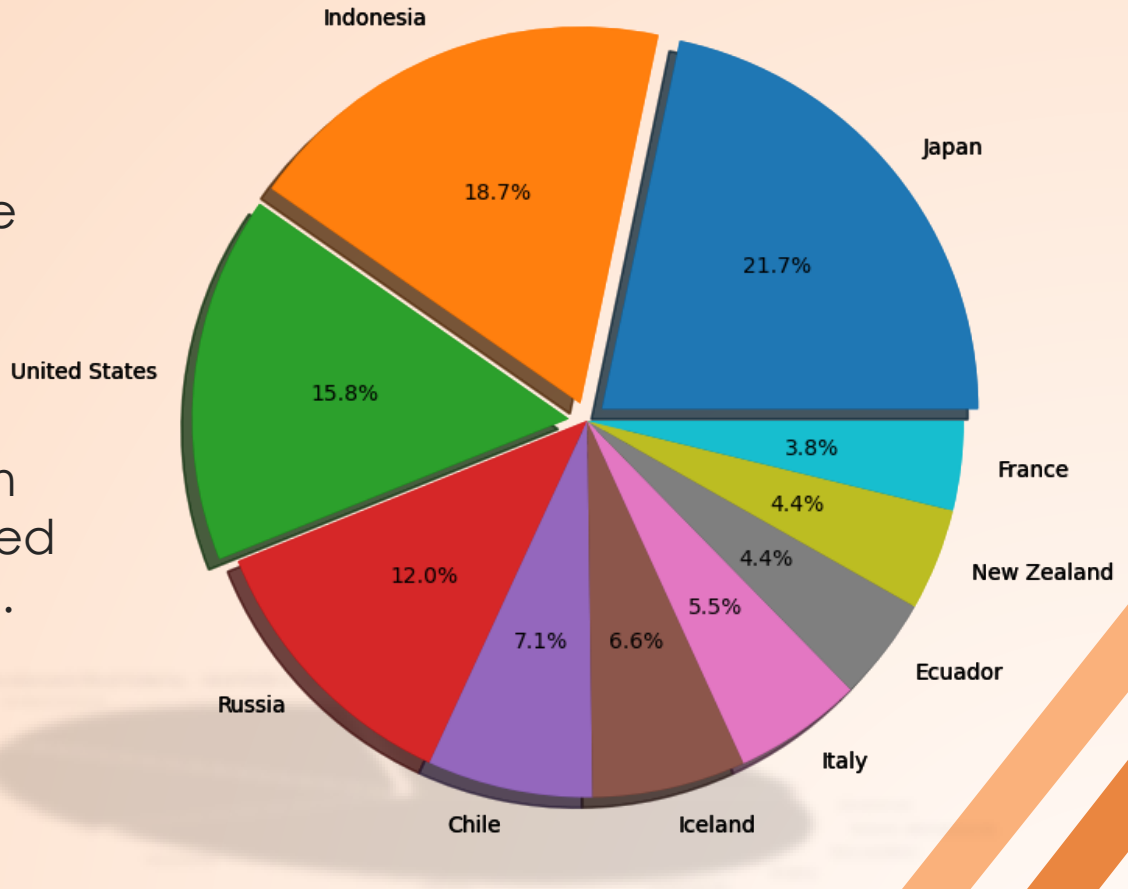
```
-4200
```


EDA

PIE CHART

Top countries

- As we can see, Japan is the country with the biggest number of eruptive volcanoes.
- On that note, Indonesia is in the second place and United States takes the third place.



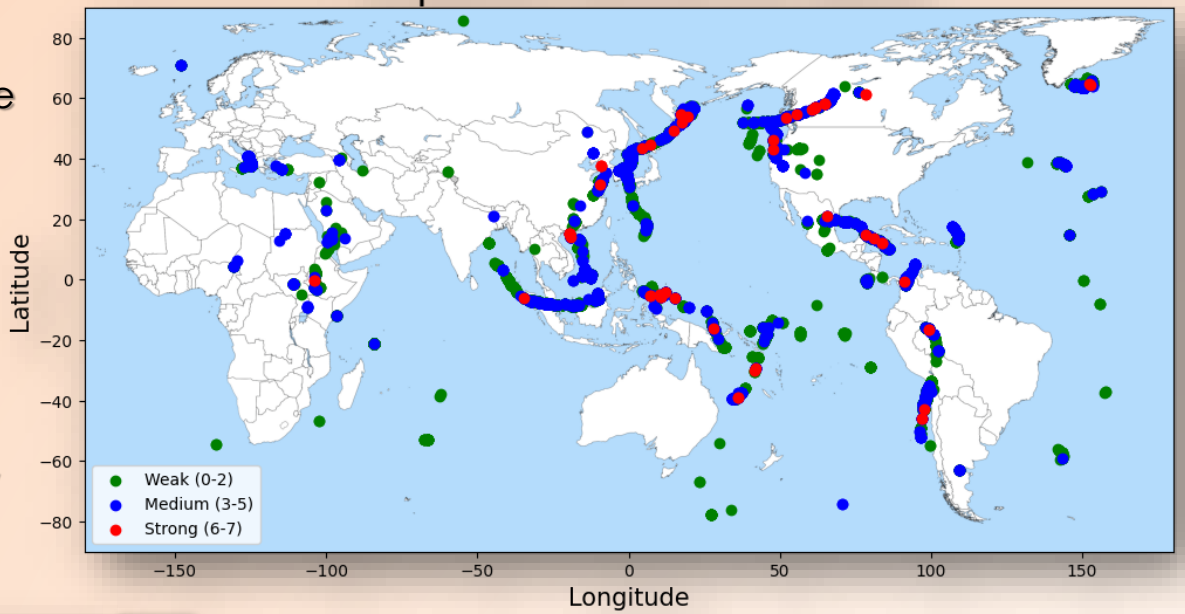
SCATTER PLOT

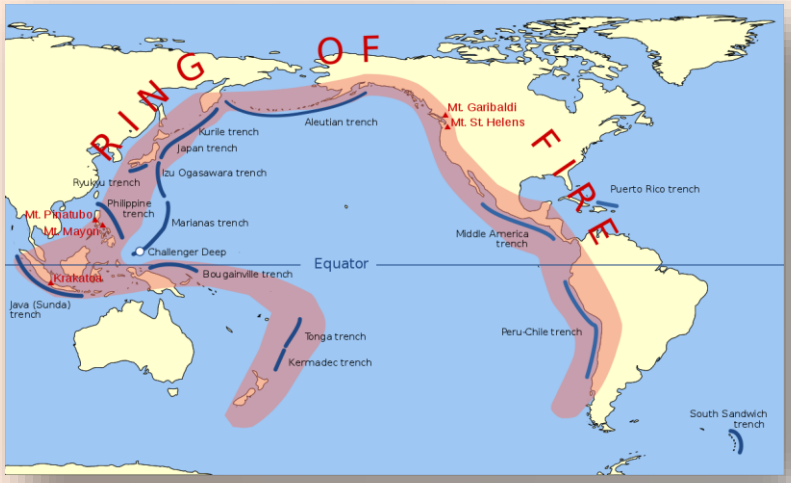
World Map

1 These are the locations of the volcanic eruptions. We can see that most of the eruptions indeed occur on the ring of fire as expected.

2 This is the ring of fire, which is a region where many volcanic eruptions occur.

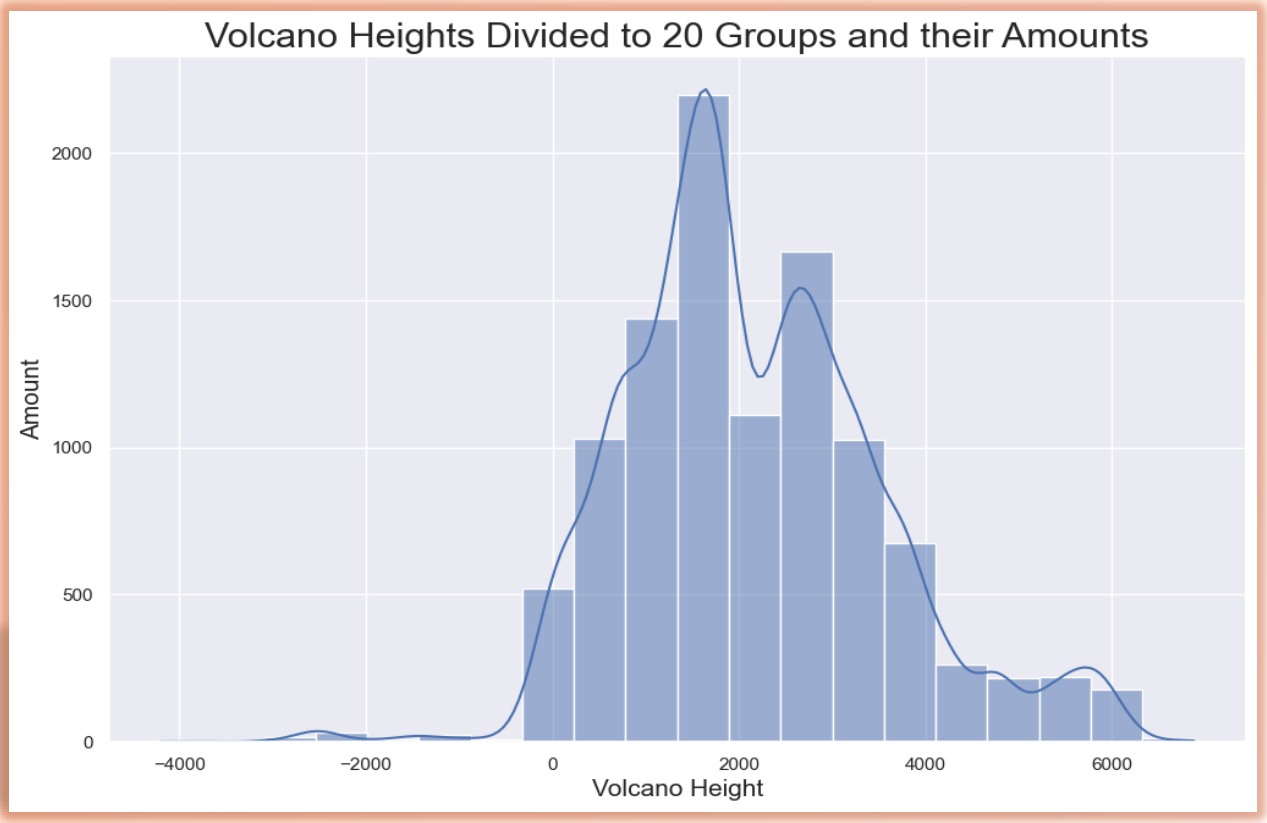
3 The ring of fire is surrounded by these continents.





HIST PLOT

- This graph shows a division of volcano heights to 20 groups.
- We can see that the most common volcano height is approximately between 1500 and 2000 meter.




MACHINE
LEARNING

KNN

KNN Using Latitude and Longtiude to Predict maxVEI

```
#### Linear Regression using latitude and longtiude , to predict maxVEI
X = df[["latitude","longitude"]]
y = df[["maxVEI"]]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

parameters = {'n_neighbors':range(1,int(math.sqrt(df.shape[0])),2) }
knn = KNeighborsClassifier()
model = GridSearchCV(knn, parameters,scoring=make_scorer(metrics.accuracy_score, greater_is_better=True))
model.fit(X_train, y_train)

print("Best parameter set is:",model.best_params_," and its score was",(round(model.best_score_,4)))
```

```
Best parameter set is: {'n_neighbors': 17}  and its score was 0.5996
```

```
userinput='y'

while(userinput=='y' or userinput=="Y"):
    lat = random.uniform(-90, 90)
    long = random.uniform(-180, 180)
    print("\nThe predicted VEI at position ("+str(round(lat,2))+","+str(round(long,2))+") is "+str(int(model.predict([[lat, long]]))))
    userinput=str(input("Try again? Y/N:  "))
print("\nDone")
```

```
The predicted VEI at position (19.1,2.76) is 2
Try again? Y/N:  y

The predicted VEI at position (65.73,95.55) is 4
Try again? Y/N:  y

The predicted VEI at position (-34.54,-133.02) is 0
Try again? Y/N:  y
```

LOGISTIC REGRESSION

Logistic Regression to Predict Eruption Strength Based on Location

```
strongEruptions = df[(df['maxVEI']>=3)]
weakEruptions = df[(df['maxVEI']<=2)]

strongEruptions['veiGroup'] = 'Strong'
weakEruptions['veiGroup'] = 'Weak'

newDF_grouped_by_maxVEI = pd.concat([weakEruptions, strongEruptions], ignore_index=True).drop(['maxVEI'], axis=1)

X = newDF_grouped_by_maxVEI[["latitude","longitude"]]
y = newDF_grouped_by_maxVEI["veiGroup"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=40)
scaler = MinMaxScaler(feature_range=(-1,1))
X_train_scaled = scaler.fit_transform(X_train.values)
model = sklearn.linear_model.LogisticRegression().fit(X_train_scaled, y_train.values)


scaler.fit(X_test)

for i in range(0,2):
    if i==0:
        print("First location will be (0,112) which is above Australia , right on the Ring of Fire")
        lat=0
        long = 112
    else:
        print("\n\nSecond location will be (90,135) - Antarctica , which has volcanoes but is relatively more quiet")
        lat=90
        long = 135

    position = scaler.transform([[lat, long]])

    weak, strong = model.predict_proba(position)[0][0], model.predict_proba(position)[0][1]
    print("\nWeak eruption probability = "+str(round(weak,4)*100)+"% \nStrong eruption probability = "+str(round(strong,4)*100)+"%\n")

print("\nThe results are as expected! \na location on the 'Ring of Fire' has a much higher 'Strong eruption probability'")
```

```
First location will be (0,112) which is above Australia , right on the Ring of Fire

Weak eruption probability = 20.77%
Strong eruption probability = 79.23%
```

Multiple ML algos

```
print("Classifier = 'KNN':")
parameters = {'n_neighbors':range(3,103,2) }
knn = KNeighborsClassifier()
clf = GridSearchCV(knn, parameters,scoring=make_scorer(metrics.accuracy_score, greater_is_better=True))
clf.fit(X_train, y_train)
params_KNN = list(clf.best_params_.values())[0]
print("best parameter set is:",clf.best_params_," and its score was",clf.best_score_)
print("-------------------------------------------")

print("Classifier = 'Decision Tree':")
parameters = {'max_depth':range(1,20),'min_samples_split':range(10,50,5) }
decisionTree = tree.DecisionTreeClassifier()
clf = GridSearchCV(decisionTree, parameters,scoring=make_scorer(metrics.accuracy_score, greater_is_better=True))
clf.fit(X_train, y_train)
params_DT_max_depth = list(clf.best_params_.values())[0]
params_DT_min_samples_split = list(clf.best_params_.values())[1]
print("best parameter set is:",clf.best_params_," and its score was",clf.best_score_)
print("-------------------------------------------")

print("Classifier = 'Random Forest':")
parameters = {'n_estimators':range(50,551,50) }
rf = RandomForestClassifier()
clf = GridSearchCV(rf, parameters,scoring=make_scorer(metrics.accuracy_score, greater_is_better=True))
clf.fit(X_train, y_train)
params_RF = list(clf.best_params_.values())[0]
print("best parameter set is:",clf.best_params_," and its score was",clf.best_score_)
print("-------------------------------------------")
```

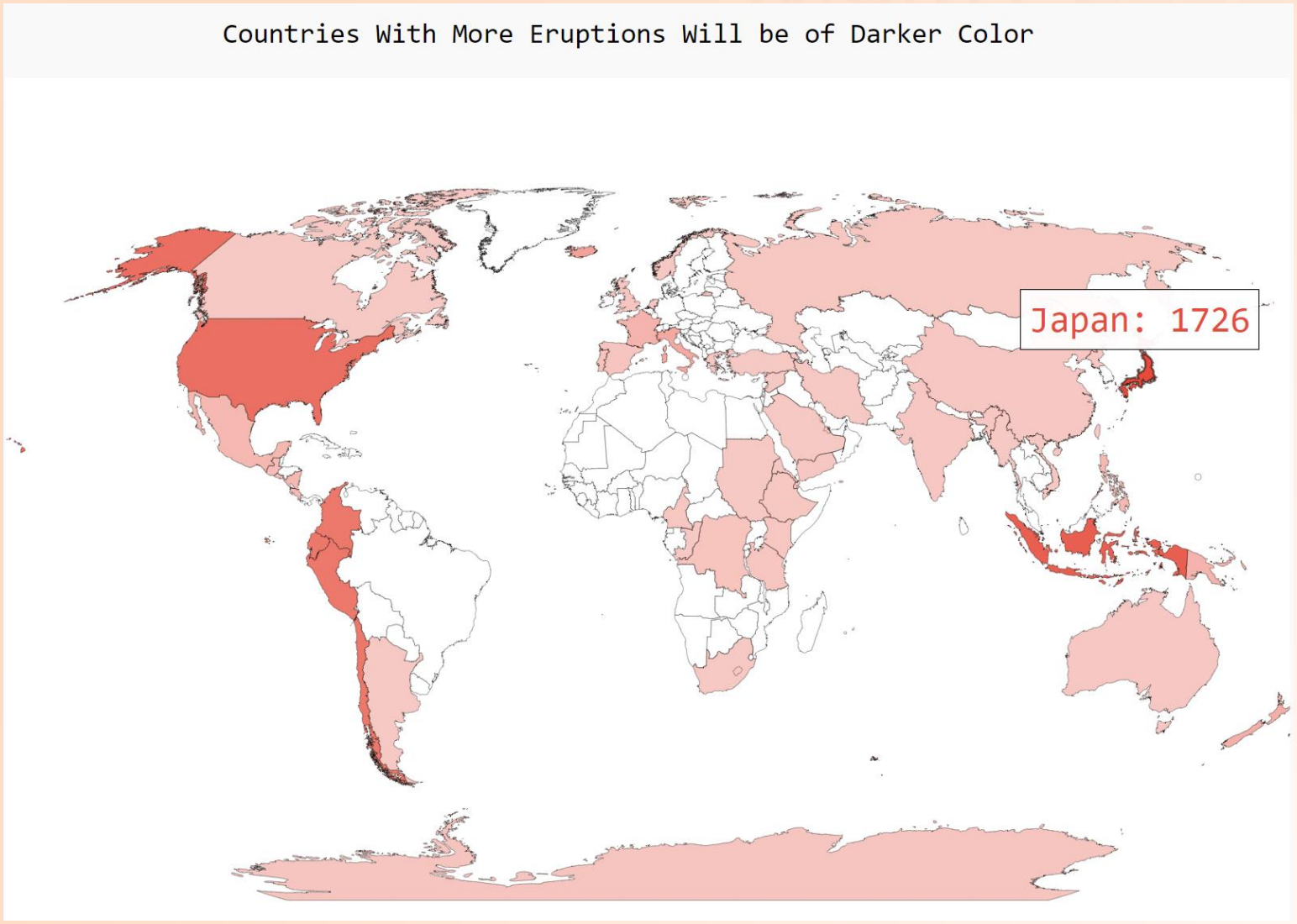
```
Enter volcano height: -2500
The predicted primary volcano type is: ['Submarine']
Try again? Y/N:  y
Enter volcano height: 1500
The predicted primary volcano type is: ['Shield']
Try again? Y/N:  y
Enter volcano height: 3500
The predicted primary volcano type is: ['Volcanic field']
Try again? Y/N:  n

Done
```

```
clf1 = KNeighborsClassifier(n_neighbors=params_KNN)
clf2 = tree.DecisionTreeClassifier(max_depth= params_DT_max_depth, min_samples_split= params_DT_min_samples_split)
clf3 = RandomForestClassifier(n_estimators=params_RF,bootstrap=True)
clf4 = GaussianNB()
algNames=["KNN", "Decision Tree", "Random Forest", "Naive Bayes"]
bestCLF = None
bestACC = 0
for i,clf in enumerate([clf1,clf2,clf3,clf4]):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    y_pred_train = clf.predict(X_train)
    print(algNames[i])
    currentACC = metrics.accuracy_score(y_true = y_train, y_pred = y_pred_train)
    print('Accuracy on training data = ', metrics.accuracy_score(y_true = y_train, y_pred = y_pred_train))
    print('Accuracy on test data = ', metrics.accuracy_score(y_true = y_test, y_pred = y_pred))
    print("-------------------------------------------")
    if currentACC > bestACC:
        bestACC = currentACC
        bestCLF=clf
print("The best CLF is: "+str(bestCLF))
```

```
The best CLF is: RandomForestClassifier(n_estimators=400)
```


WOW
EFFECT

Wow Effect


Countries With More Eruptions Will be of Darker Color
Japan: 1726


CONCLUSIONS

CONCLUSIONS

Conclusions

We learned a lot about volcanic eruptions, observing and plotting the data teached us many interesting things , for example , that Japan is the leading country when it comes to eruptions amount. We learned how powerful machine learning can be, and how useful it can be when trying to find patterns in the data. We were able to predict the strength of the next eruption, as well as predicting volcano types based on their heights and also s02 emissions.

THANK YOU!