

Meteorite Landings: Patterns and Predictions

ICCS 261 Term Project

By: Parit Vacharaskunee (6580209)

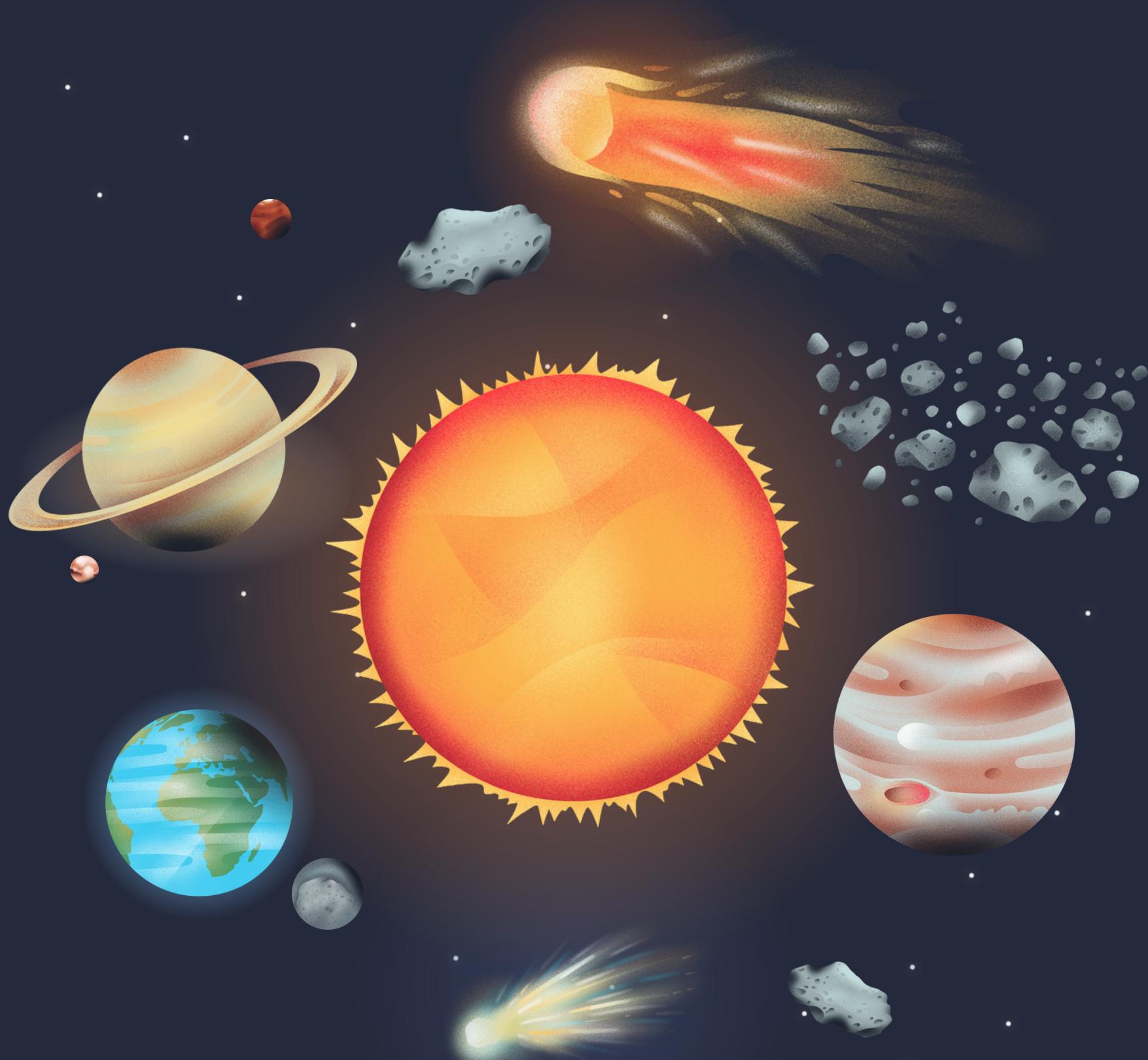


Table of Contents



- 1. Introduction and Research Question**
- 2. Purpose and Objectives**
- 3. Data Cleaning and Preparation**
 - a. Classification of Meteorites**
- 4. EDA: Proposal Reflection**
- 5. Fitting and Evaluating Models**
- 6. Interpretation and Analysis**
- 7. Conclusion**

Introduction

- Meteorite landings are intriguing natural phenomena that gives us **insights** on space.
- Meteorology provides clues about the early solar system and the **formation** of planets.
- Aims to explore **patterns** in meteorite landings and develop **classification models**.

Research Question: Can meteorites be distinguished to either stone, iron, or stony iron types by its features?

Purpose and Objectives

Purpose: Gain insights into **patterns** and **characteristics** that distinguish types of meteorites. It could enhance **understanding** of the classification of meteorites.

Objectives: Determine **features** to be used in classifying meteorites by performing **EDA's** on data. Develop **classification models** with different parameters. Evaluate models and conclude.

Data Cleaning/Preparation

Data obtained from: [NASA's Meteorite Landings Dataset](#)

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

	name	id	nametype	recclass	mass (g)	fall	year	reclat	reclong	GeoLocation
0	Aachen	1	Valid	L5	21.0	Fell	1880.0	50.77500	6.08333	(50.775, 6.08333)
1	Aarhus	2	Valid	H6	720.0	Fell	1951.0	56.18333	10.23333	(56.18333, 10.23333)
2	Abee	6	Valid	EH4	107000.0	Fell	1952.0	54.21667	-113.00000	(54.21667, -113.0)
3	Acapulco	10	Valid	Acapulcoite	1914.0	Fell	1976.0	16.88333	-99.90000	(16.88333, -99.9)
4	Achiras	370	Valid	L6	780.0	Fell	1902.0	-33.16667	-64.95000	(-33.16667, -64.95)
...
45711	Zillah 002	31356	Valid	Eucrite	172.0	Found	1990.0	29.03700	17.01850	(29.037, 17.0185)
45712	Zinder	30409	Valid	Pallasite, ungrouped	46.0	Found	1999.0	13.78333	8.96667	(13.78333, 8.96667)
45713	Zlin	30410	Valid	H4	3.3	Found	1939.0	49.25000	17.66667	(49.25, 17.66667)
45714	Zubkovsky	31357	Valid	L6	2167.0	Found	2003.0	49.78917	41.50460	(49.78917, 41.5046)
45715	Zulu Queen	30414	Valid	L3.7	200.0	Found	1976.0	33.98333	-115.68333	(33.98333, -115.68333)

45716 rows × 10 columns

Drop Columns and Checking Missing Values

1.

```
df = df.drop(['id', 'nametype', 'GeoLocation'], axis = 1)
df.isna().sum()

✓ 0.0s
```

name	0
recclass	0
mass (g)	131
fall	0
year	291
reclat	7315
reclong	7315
dtype:	int64

3.



	name	recclass	mass (g)	fall	year	reclat	reclong
0	Aachen	L5	21.0	Fell	1880.0	50.77500	6.08333
1	Aarhus	H6	7.00	Fell	1951.0	56.18333	10.23333
2	Abee	EH4	107000.0	Fell	1952.0	54.21667	-113.00000
3	Acapulco	Acapulcoite	1914.0	Fell	1976.0	16.88333	-99.90000
4	Achira	L6	780.0	Fell	1902.0	-33.16667	-64.95000
...
45711	Zillah	Euclite	172.0	Found	1990.0	29.03700	17.01850
45712	Zinder	Pallasite, ungrouped	46.0	Found	1999.0	13.78333	8.96667
45713	Zlin	H4	3.0	Found	1939.0	49.25000	17.66667
45714	Zubkovsky	L6	216.0	Found	2003.0	49.78917	41.50460
45715	Zulu Queen	L3.7	200.0	Found	1976.0	33.98333	-115.68333

45716 rows × 7 columns

2.

```
# Since there are a lot of missing data in reclat/reclong, a forward fill will be done as no significant average can be calculated from locations.
df[['reclat', 'reclong']] = df[['reclat', 'reclong']].fillna(method = 'ffill')

✓ 0.0s
```

Classification of Meteorites

Stony:

- Carbonaceous chondrites: CI; CM; CR; CO; CV; CK; CH; Ungrouped Carbonaceous chondrite
- Ordinary chondrites: H; L; LL; HH
- Rumuruti chondrites: R
- Kakangari chondrites: K
- Enstatite chondrites: EH; EL; Ungrouped enstatite chondrites
- IAB/IIICD silicate chondrites
- Ungrouped chondrites
- Primitive achondrites: Acapulcoites; Lodranites; Winonaites; Ungrouped primitive achondrites
- Differentiated meteorites:
- Asteroidal achondrites: Eucrites; Diogenites; Howardites; Angrites; Aubrites; Ureilites; Brachinites
- Martian meteorites: Shergottites; Nakhrites; Chassigny; ALH 84001
- Lunar meteorites: Mare basalts; Impact breccias

Stony irons:

- Pallasites
- Mesosiderites
- Ungrouped stony irons

Irons:

- Magmatic irons groups: IC; IIAB; IIC; IID; IIF; IIIAB; IIIE; IIIF; IVA; IVB
- Nonmagmatic irons groups: IAB/IIICD; IIE
- Ungrouped irons

Classified into 3

main types:

stones, stony-irons,
and irons

Separation by Keywords for Irons and Stony Irons

```
uniques = pd.DataFrame(df['recclass'].unique())
# retreive classes with keyword 'iron
iron = uniques[uniques[0].str.contains('Iron')]
iron = pd.concat([iron, uniques[uniques[0].str.contains('iron')]])  
  
# retreive classes with keywords regarding stony irons
stony_iron = uniques[uniques[0].str.contains('Pallasite')]
stony_iron = pd.concat([stony_iron, uniques[uniques[0].str.contains('Mesosiderite')]])  
  
# dropping out irons and stony irons from the main dataset
stone = uniques.drop(iron.index).drop(stony_iron.index)  
✓ 0.0s
```

```
# renaming the names in recclass. WARNING: if cell is run twice, do a run-all. It messes up the variables of iron and stony iron.
df['recclass'] = df['recclass'].replace(iron.values, 'Iron')
df['recclass'] = df['recclass'].replace(stony_iron.values, 'Stony Iron')
df['recclass'] = df['recclass'].replace(stone.values, 'Stone')  
✓ 0.3s
```

```
df['recclass'].unique() # checking the unique names  
✓ 0.0s  
  
array(['Stone', 'Iron', 'Stony Iron'], dtype=object)
```

Final Cleaning

```
# get median for each type
med_st = df[df['recclass'] == 'Stone']['year'].median()
med_ir = df[df['recclass'] == 'Iron']['year'].median()
med_si = df[df['recclass'] == 'Stony Iron']['year'].median()

# fillna for each type
df.loc[(df['recclass'] == 'Stone') & (df['year'].isna()), 'year'] = med_st
df.loc[(df['recclass'] == 'Iron') & (df['year'].isna()), 'year'] = med_ir
df.loc[(df['recclass'] == 'Stony Iron') & (df['year'].isna()), 'year'] = med_si
```

✓ 0.0s

```
mass_st = df[df['recclass'] == 'Stone']['mass (g)'].median()
mass_ir = df[df['recclass'] == 'Iron']['mass (g)'].median()
mass_si = df[df['recclass'] == 'Stony Iron']['mass (g)'].median()

df.loc[(df['recclass'] == 'Stone') & (df['mass (g)'].isna()), 'mass (g)'] = med_st
df.loc[(df['recclass'] == 'Iron') & (df['mass (g)'].isna()), 'mass (g)'] = med_ir
df.loc[(df['recclass'] == 'Stony Iron') & (df['mass (g)'].isna()), 'mass (g)'] = med_si
```

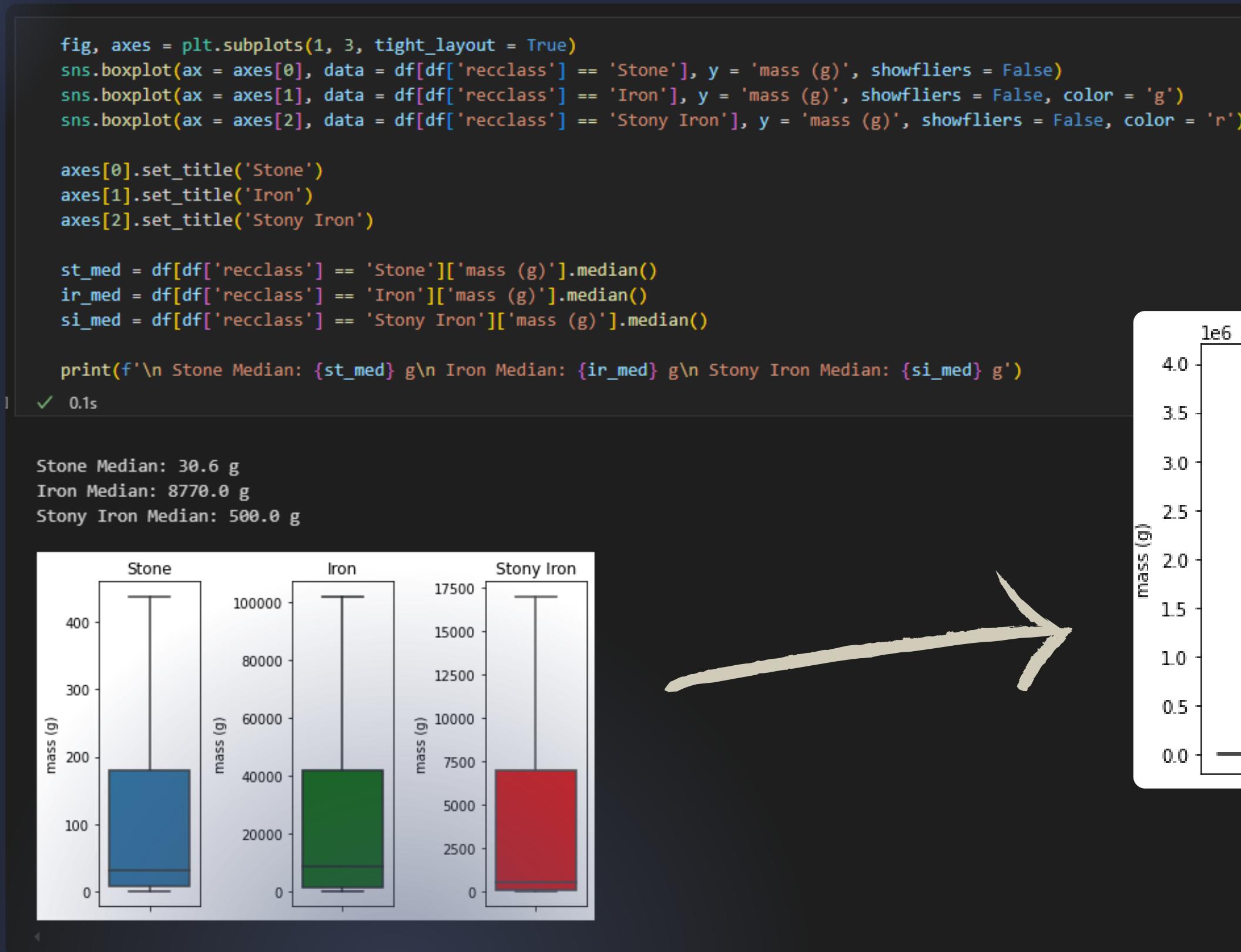
✓ 0.0s

```
df.isna().sum()
```

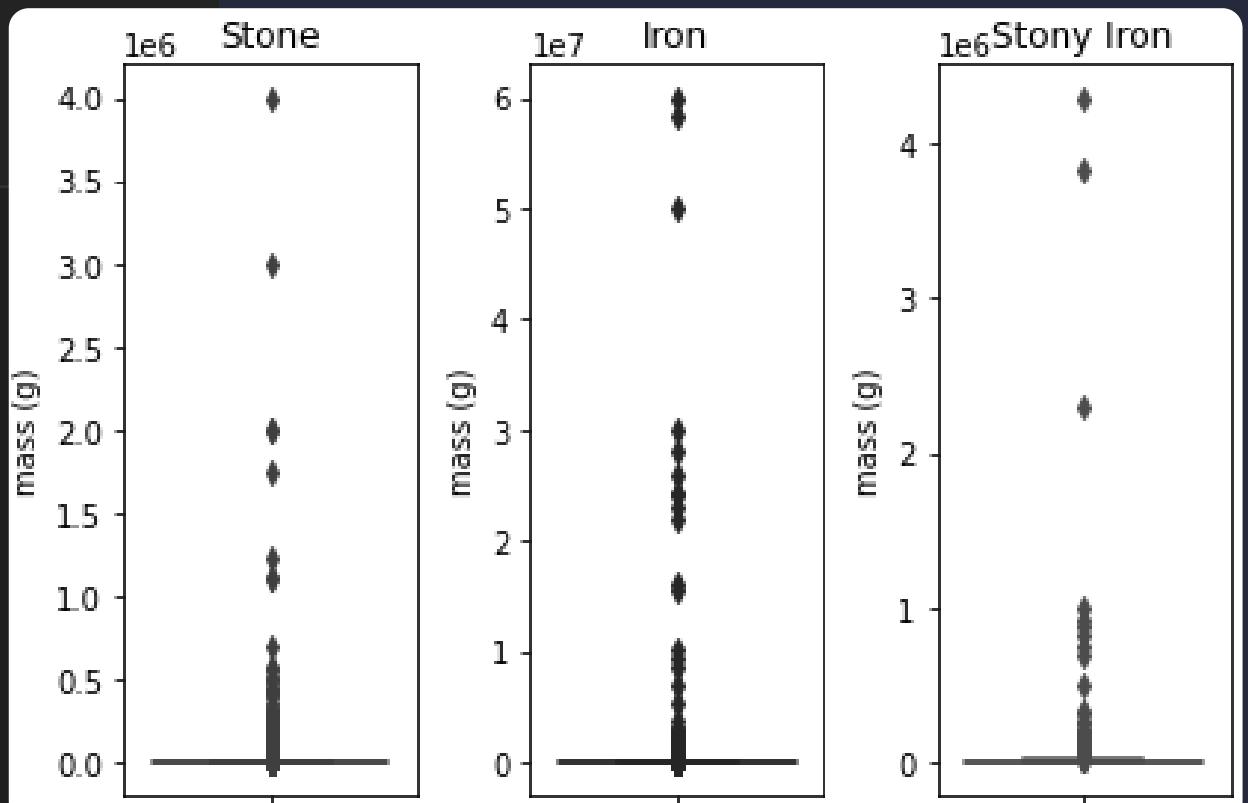
✓ 0.0s

name	0
recclass	0
mass (g)	0
fall	0
year	0
reclat	0
reclong	0
dtype: int64	

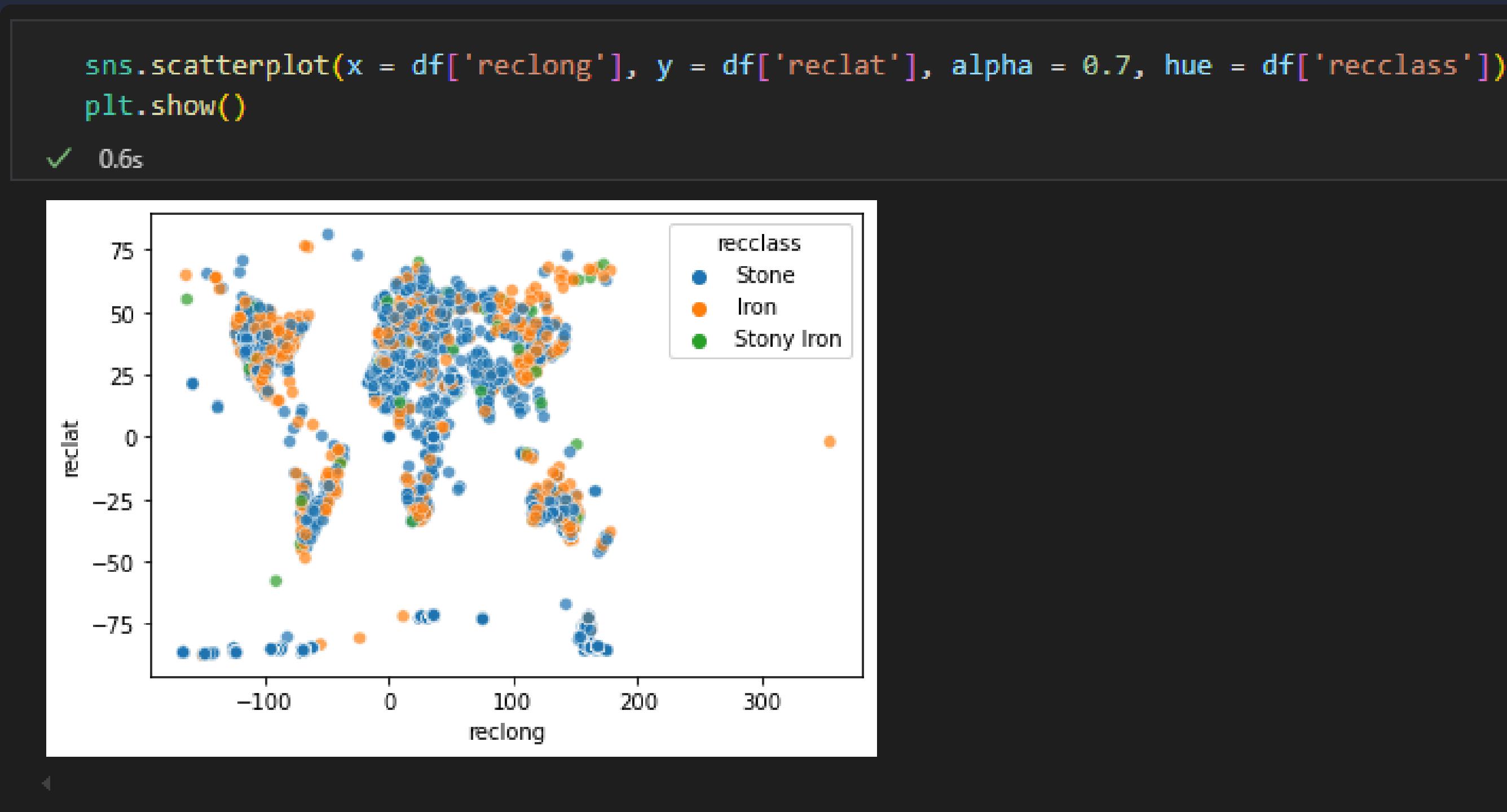
EDA: Proposal Reflection



With outliers..



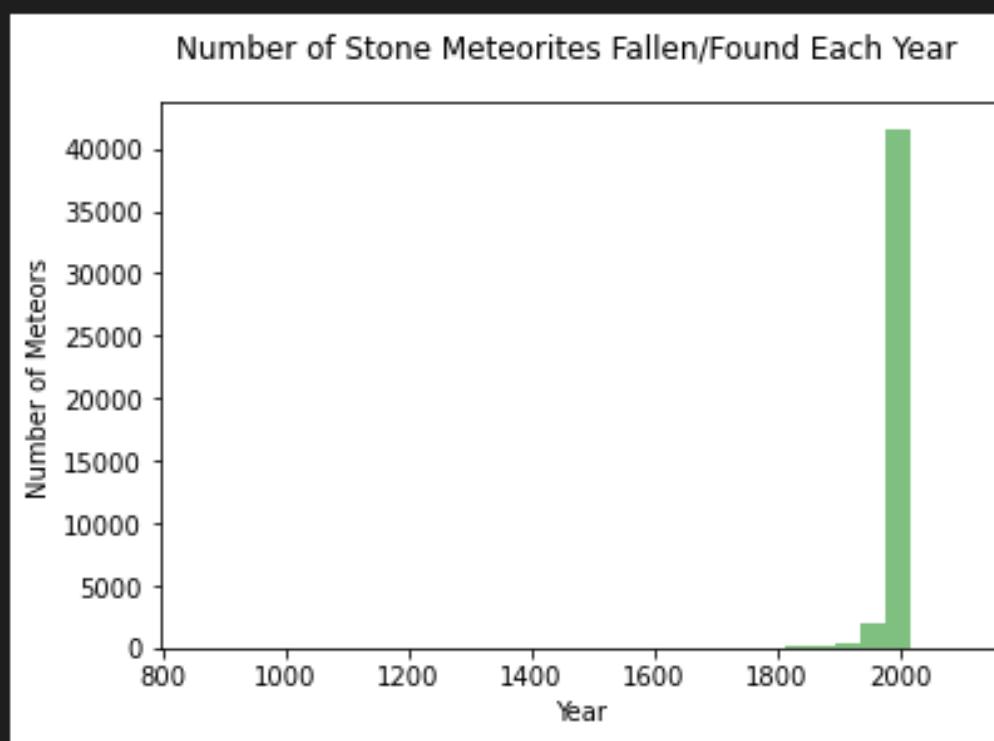
EDA: Proposal Reflection



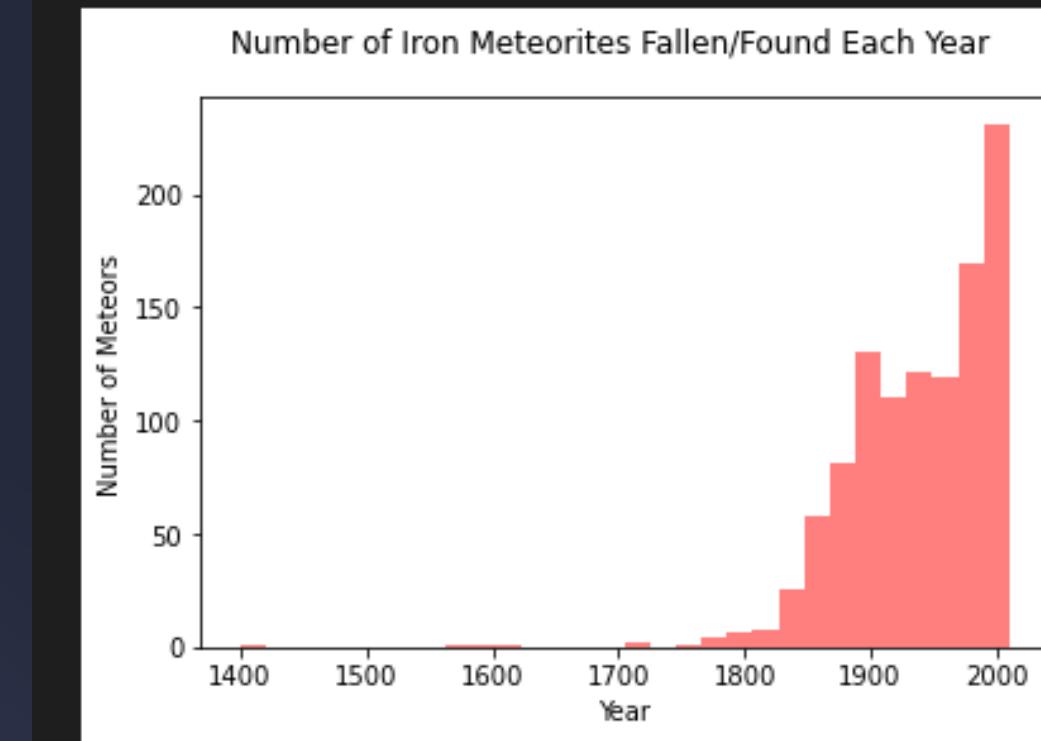
EDA: Proposal Reflection

```
# separating the years by type  
fallen_st = df[df['recclass'] == 'Stone']  
fallen_ir = df[df['recclass'] == 'Iron']  
fallen_si = df[df['recclass'] == 'Stony Iron']  
  
✓ 0.0s
```

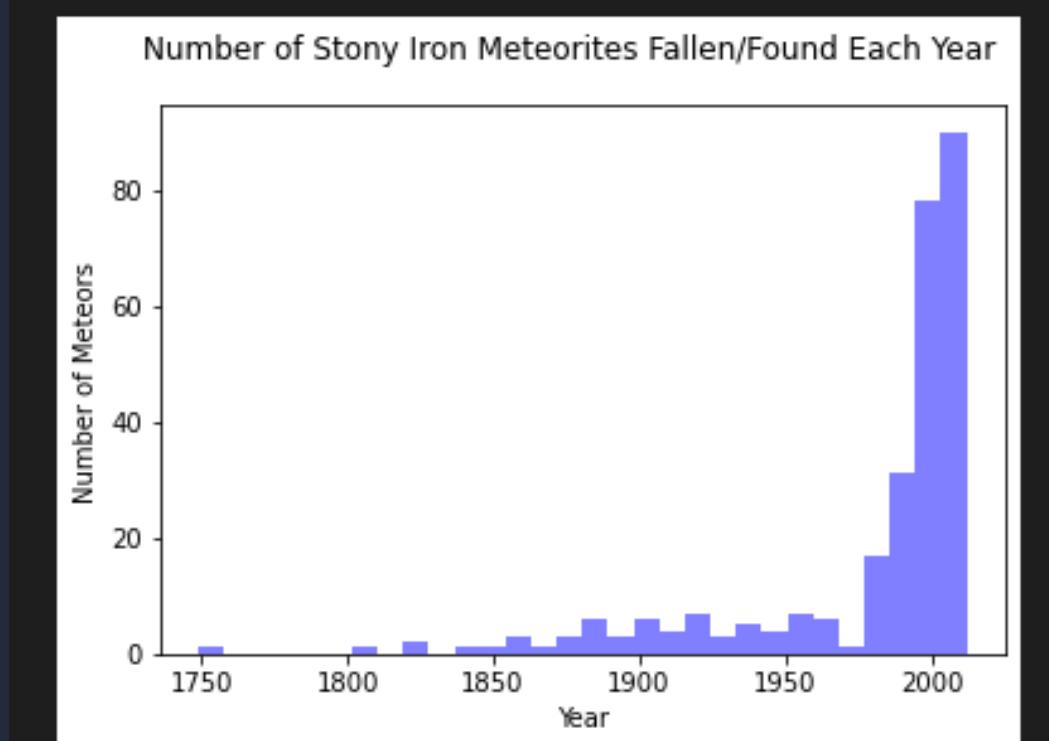
```
# counting the number of meteors in each year  
years_st = pd.DataFrame(fallen_st['year'].value_counts())  
years_ir = pd.DataFrame(fallen_ir['year'].value_counts())  
years_si = pd.DataFrame(fallen_si['year'].value_counts())  
  
✓ 0.0s
```



Median: 1998



Median: 1941



Median: 2002

Fitting and Evaluating Models:

Chosen Model: Random Forest Classifier

- Dimensionality
- Reduces Overfitting
- Feature Importance
- Non-linear

5 (4.5) models were fitted with slight changes in parameters
as an attempt to obtain better models.

Model #1: Out of the Box

1.

	mass (g)	reclat	reclong	fell	year
0	21.0	50.77500	6.08333	1	1880.0
1	720.0	56.18333	10.23333	1	1951.0
2	107000.0	54.21667	-113.00000	1	1952.0
3	1914.0	16.88333	-99.90000	1	1976.0
4	780.0	-33.16667	-64.95000	1	1902.0
...
45711	172.0	29.03700	17.01850	0	1990.0
45712	46.0	13.78333	8.96667	0	1999.0
45713	3.3	49.25000	17.66667	0	1939.0
45714	2167.0	49.78917	41.50460	0	2003.0
45715	200.0	33.98333	-115.68333	0	1976.0

```
y = df.recclass
y
✓ 0.0s

0      Stone
1      Stone
2      Stone
3      Stone
4      Stone
...
45711    Stone
45712  Stony Iron
45713    Stone
45714    Stone
45715    Stone
Name: recclass, Length: 45716, dtype: object
```

2.

```
# split train-test
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_state = 10)
✓ 0.0s

# fitting random forest model with tree size of 100
rf_model = RandomForestClassifier(n_estimators=100)
_ = rf_model.fit(xtrain, ytrain)
```

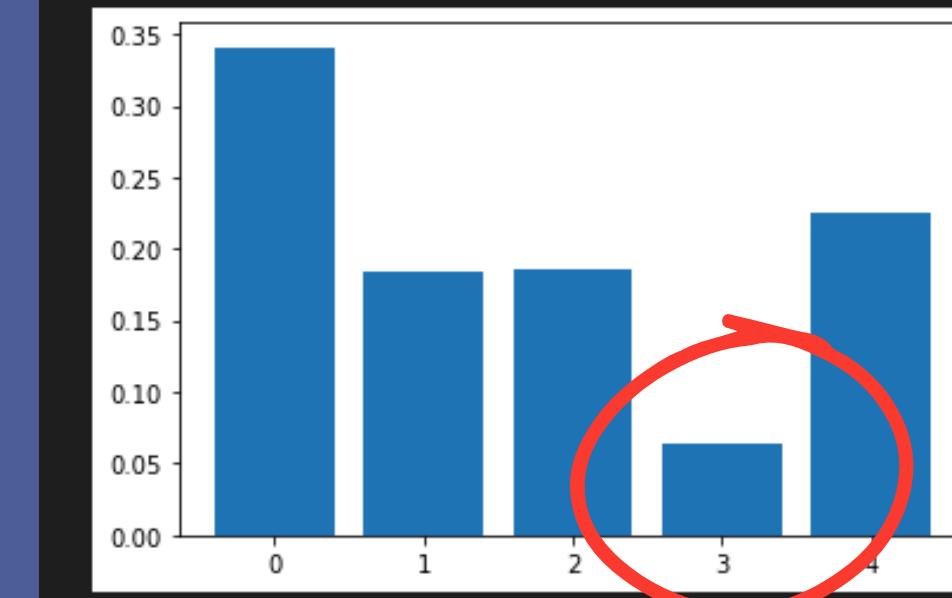
3.

```
# get importance to see which variables are most important when evaluating
importance = rf_model.feature_importances_

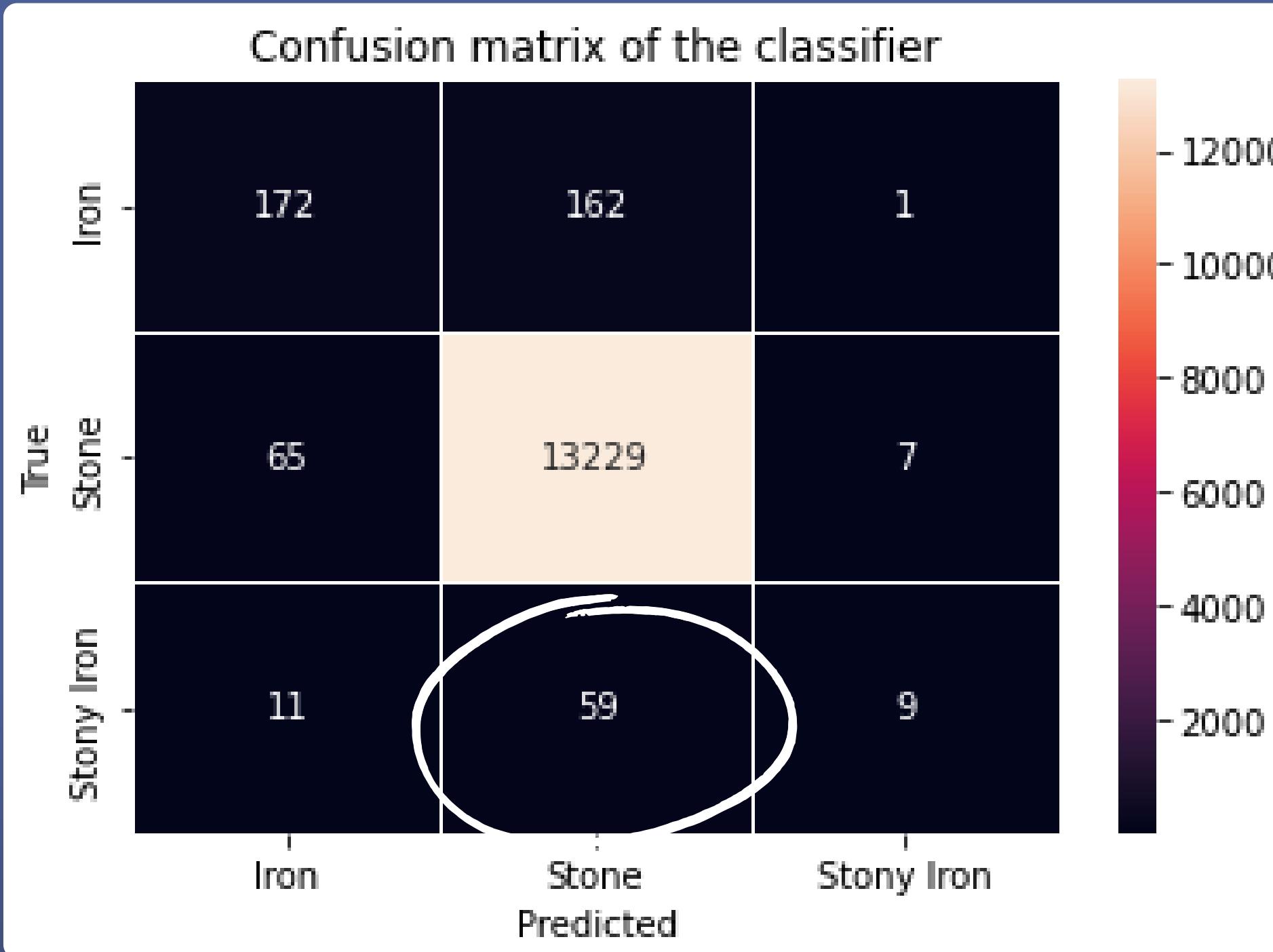
# summarize feature importance
for i,v in enumerate(importance):
| print('Feature: %d, Score: %.5f' % (i,v))

# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
✓ 2.6s

Feature: 0, Score: 0.34080
Feature: 1, Score: 0.18394
Feature: 2, Score: 0.18631
Feature: 3, Score: 0.06326
Feature: 4, Score: 0.22568
```



Confusion matrix of the classifier

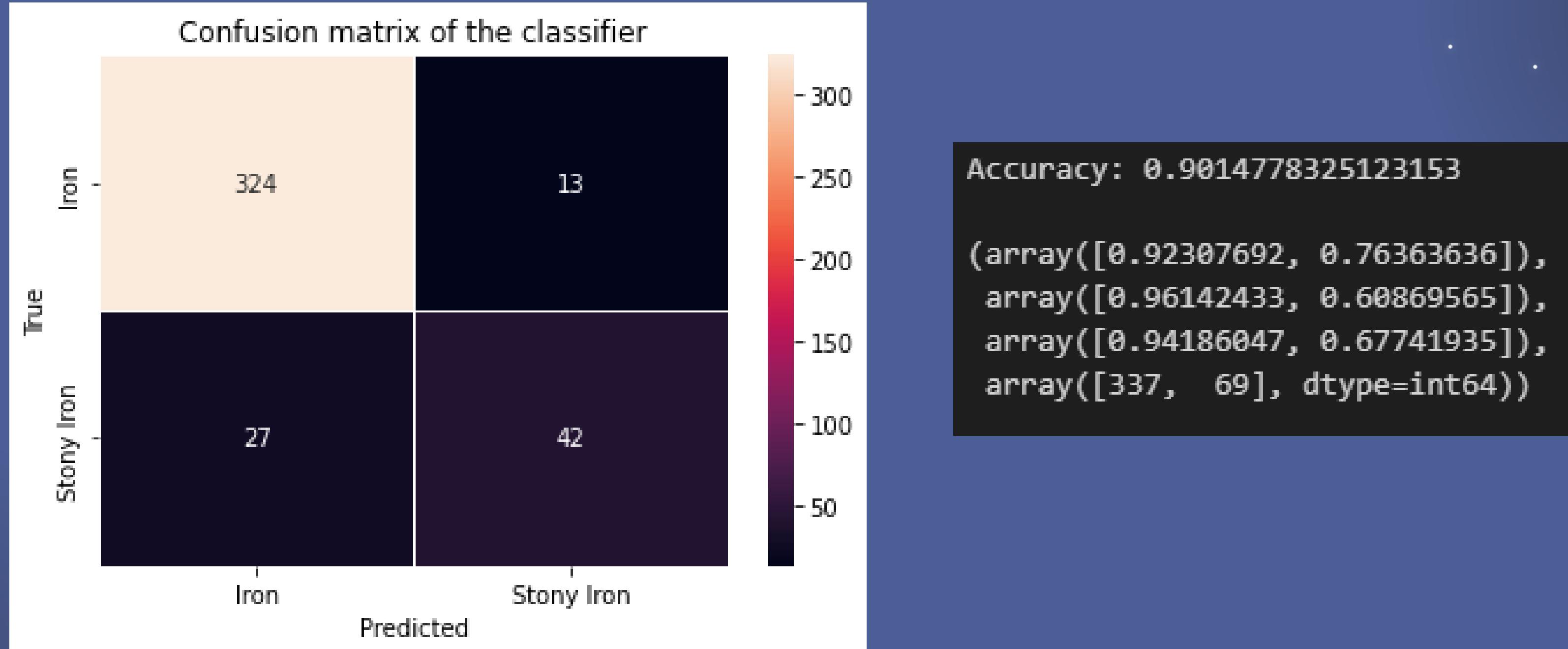


Accuracy: 0.977761574917973

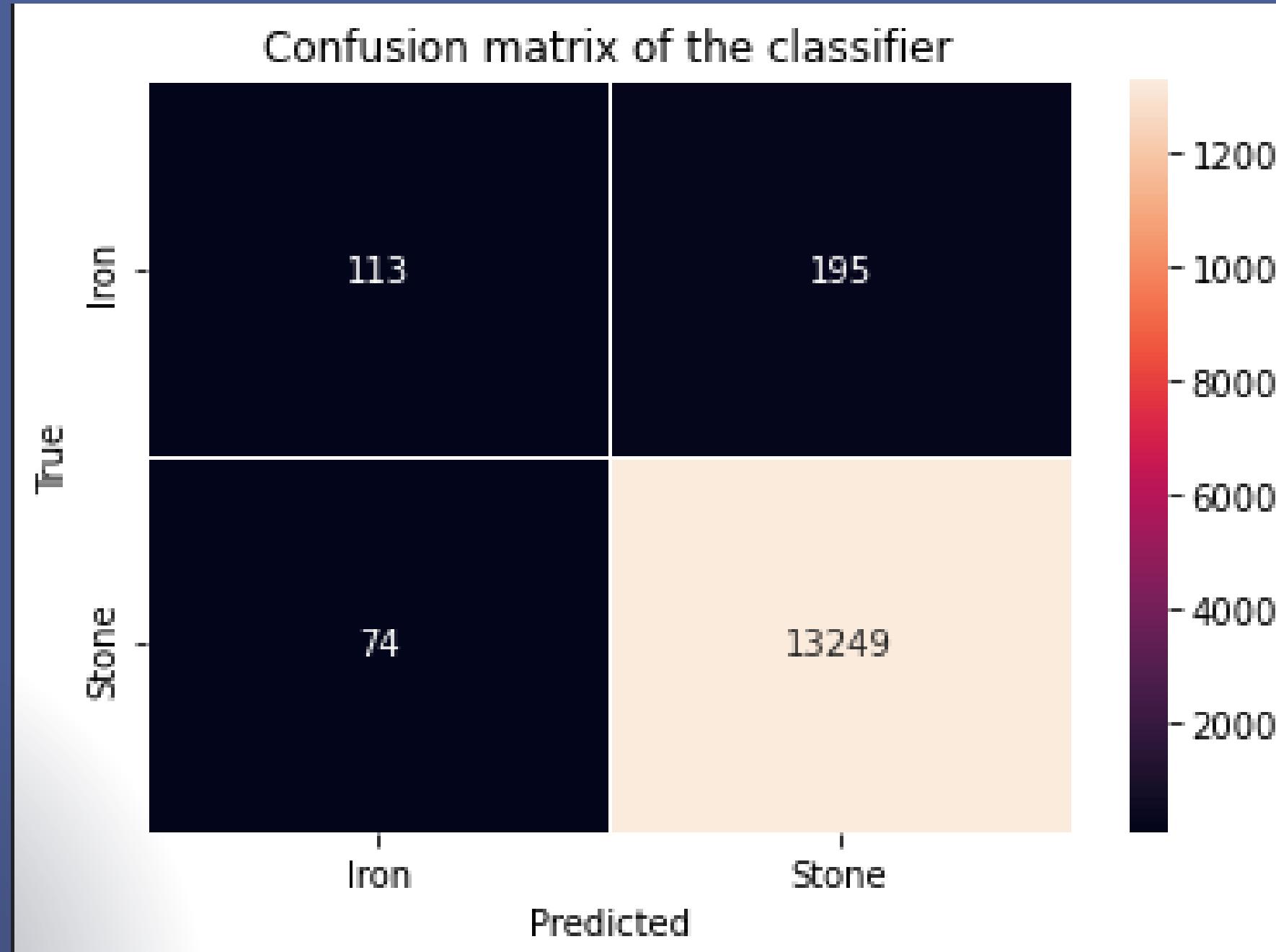
(array([0.69354839, 0.98356877, 0.52941176]),
 array([0.51343284, 0.99458687, 0.11392405]),
 array([0.59005146, 0.98904714, 0.1875]),
 array([335, 13301, 79], dtype=int64))

Stony irons and stones are hard to separate (?)

Model #2: Stony Irons vs. Irons



Model #3: Stones vs. Irons



```
Accuracy: 0.9802655711246424  
(array([0.60427807, 0.98549539]),  
 array([0.36688312, 0.9944457 ]),  
 array([0.45656566, 0.98995031]),  
 array([ 308, 13323], dtype=int64))
```

Model #4: Weights + No Outliers

1.

```
# stones
stone_sub = df[df['recclass'] == 'Stone']
iron_sub = df[df['recclass'] == 'Iron']
stonyiron_sub = df[df['recclass'] == 'Stony Iron']

# calculate IQR
stone_IQR = stone_sub['mass (g)'].quantile(0.75) - stone_sub['mass (g)'].quantile(0.25)
iron_IQR = iron_sub['mass (g)'].quantile(0.75) - iron_sub['mass (g)'].quantile(0.25)
stonyiron_IQR = stonyiron_sub['mass (g)'].quantile(0.75) - stonyiron_sub['mass (g)'].quantile(0.25)

# get outliers
stone_outliers = stone_sub.loc[(stone_sub['mass (g)'] < (stone_sub['mass (g)'].quantile(0.25) - 1.5 * stone_IQR)) | (df['mass (g)'] > (stone_sub['mass (g)'].quantile(0.75) + 1.5 * stone_IQR))]
iron_outliers = iron_sub.loc[(iron_sub['mass (g)'] < (iron_sub['mass (g)'].quantile(0.25) - 1.5 * iron_IQR)) | (df['mass (g)'] > (iron_sub['mass (g)'].quantile(0.75) + 1.5 * iron_IQR))]
stonyiron_outliers = stonyiron_sub.loc[(stonyiron_sub['mass (g)'] < (stonyiron_sub['mass (g)'].quantile(0.25) - 1.5 * stonyiron_IQR)) | (df['mass (g)'] > (stonyiron_sub['mass (g)'].quantile(0.75) + 1.5 * stonyiron_IQR))]
```

2.

		name	recclass	mass (g)	fall	year	reclat	reclong
0		Aachen	Stone	21.0	Fell	1880.0	50.77500	6.08333
16		Akyumak	Iron	50000.0	Fell	1981.0	39.91667	42.81667
17		Al Rais	Stone	160.0	Fell	1957.0	24.41667	39.51667
22		Alby sur Chéran	Stone	252.0	Fell	2002.0	45.82133	6.01533
37		Northwest Africa 5815	Stone	256.8	Found	1998.0	0.00000	0.00000
...	
45709		Zhongxiang	Iron	100000.0	Found	1981.0	31.20000	112.50000
45711		Zillah 002	Stone	172.0	Found	1990.0	29.03700	17.01850
45712		Zinder	Stony Iron	46.0	Found	1999.0	13.78333	8.96667
45713		Zlin	Stone	3.3	Found	1939.0	49.25000	17.66667
45715		Zulu Queen	Stone	200.0	Found	1976.0	33.98333	-115.68333

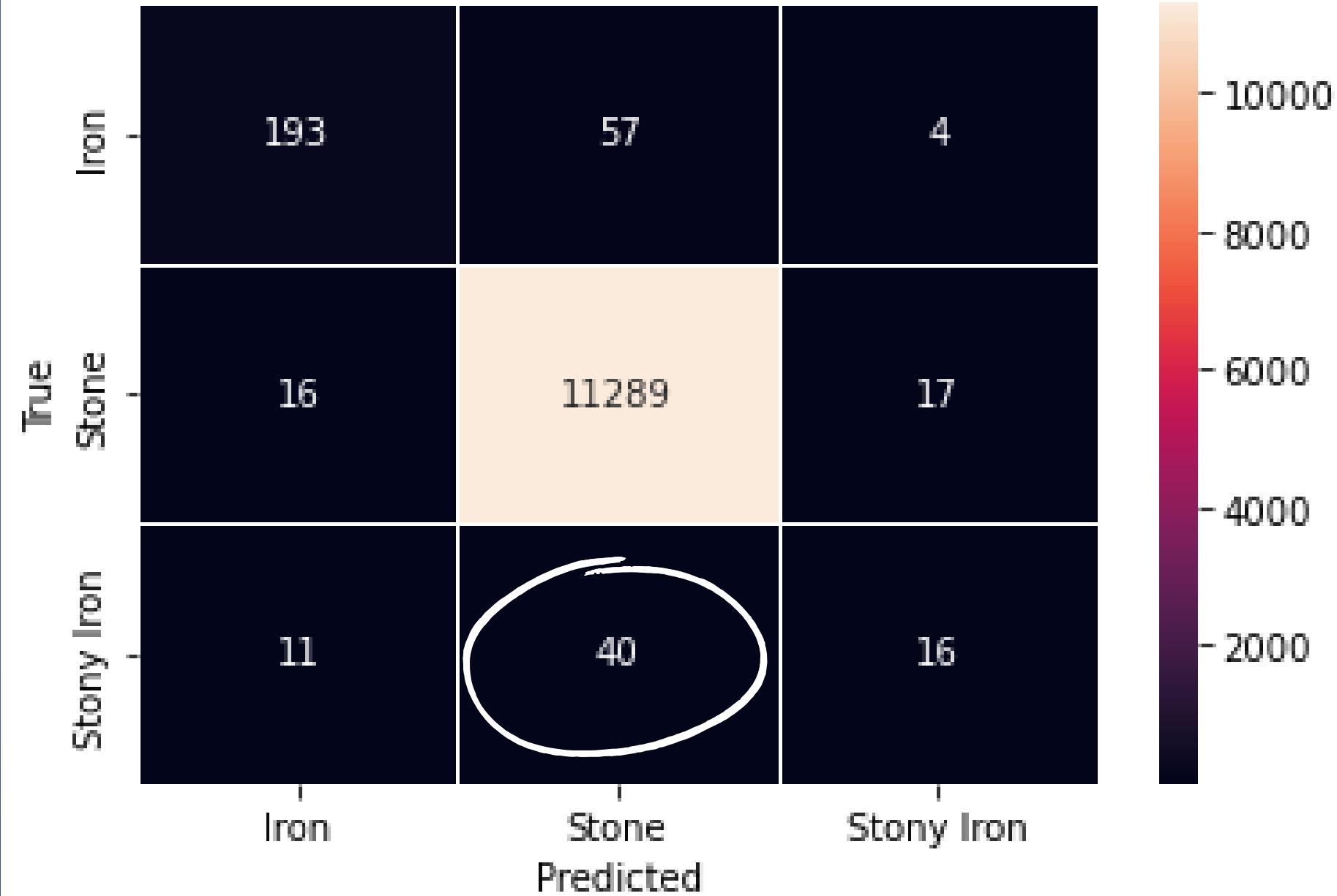
38808 rows × 7 columns

3.

```
# fitting random forest model with tree size of 100
rf_model4 = RandomForestClassifier(n_estimators=100, class_weight='balanced')
_ = rf_model4.fit(xtrain4, ytrain4)

✓ 1.9s
```

Confusion matrix of the classifier



Accuracy: 0.9875461650777291

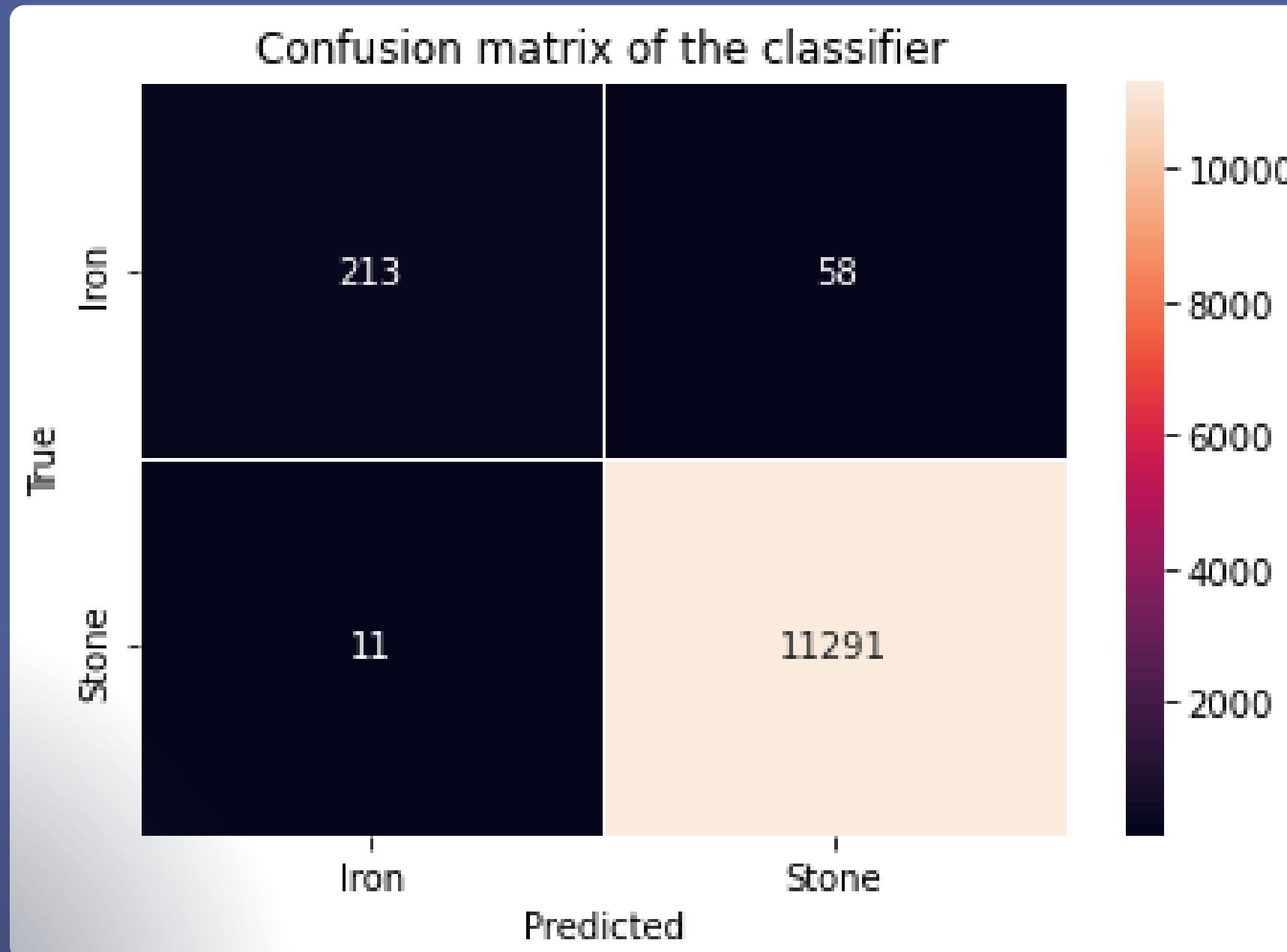
```
(array([0.87727273, 0.99148077, 0.43243243]),  
 array([0.75984252, 0.99708532, 0.23880597]),  
 array([0.81434599, 0.99427515, 0.30769231]),  
 array([-254, 11322, 67], dtype=int64))
```

From **model #1** for comparison

Accuracy: 0.977761574917973

```
(array([0.69354839, 0.98356877, 0.52941176]),  
 array([0.51343284, 0.99458687, 0.11392405]),  
 array([0.59005146, 0.98904714, 0.1875]),  
 array([-335, 13301, 79], dtype=int64))
```

Model #4.5: Revisiting Model #3



```
Accuracy: 0.9940378467121749  
  
(array([0.95089286, 0.99488942]),  
 array([0.78597786, 0.99902672]),  
 array([0.86066686, 0.99695378]),  
 array([-271, 11302], dtype=int64))
```

Interpretation and Analysis

- **Stone meteorites** have a big impact on the **accuracy** score.
- **Model 2** scored best for classification between **stony-irons** and **irons**.
- **Model 4.5** scored best for classification between **stone** and **irons**.
- **Weight** is the most significant feature to help classify.

Question: Is it optimal to remove outliers?

Conclusion

- Regarding the **research question**, models from this project indicate that it is possible to classify **meteorites** to their specific types by their **characteristics** and **features**.
- The question of whether or not these models are **optimal** and **nonbiased** depends on the individual.
- Research is only limited to classifying the **main types**, not sub-types.
 - Requires further **observation** and **investigation** by scientists/researchers.