data_624_homework_four_Kevin_Kirby

September 23, 2024

1 Assignment Four for Fall 2024 DATA 624: Preprocessing and Overfitting

Author: Kevin Kirby This assignment consists of exercises 3.1 and 3.2 from Applied Predictive Modeling by Max Kuhn and Kjell Johnson. Each question is as provided in the book and has the pandas library required for that section that aren't already imported.

1.1 3.1. UC Irvine Machine Learning Repository Glass Identification Review

Information about the dataset can be found here. I imported it using Python code provided by ICU. Per the exercise question, "the data consist of 214 glass samples labeled as one of seven class categories. There are nine predictors, including the refractive index and percentages of eight elements: Na, Mg, Al, Si, K, Ca, Ba, and Fe."

The exercise asks for the following:

A. Using visualizations, explore the predictor variables to understand their distributions as well as the relationships between predictors.
B. Do there appear to be any outliers in the data? Are any predictors skewed?
C. Are there any relevant transformations of one or more predictors that might improve the classification model?

First, import and setup the data:

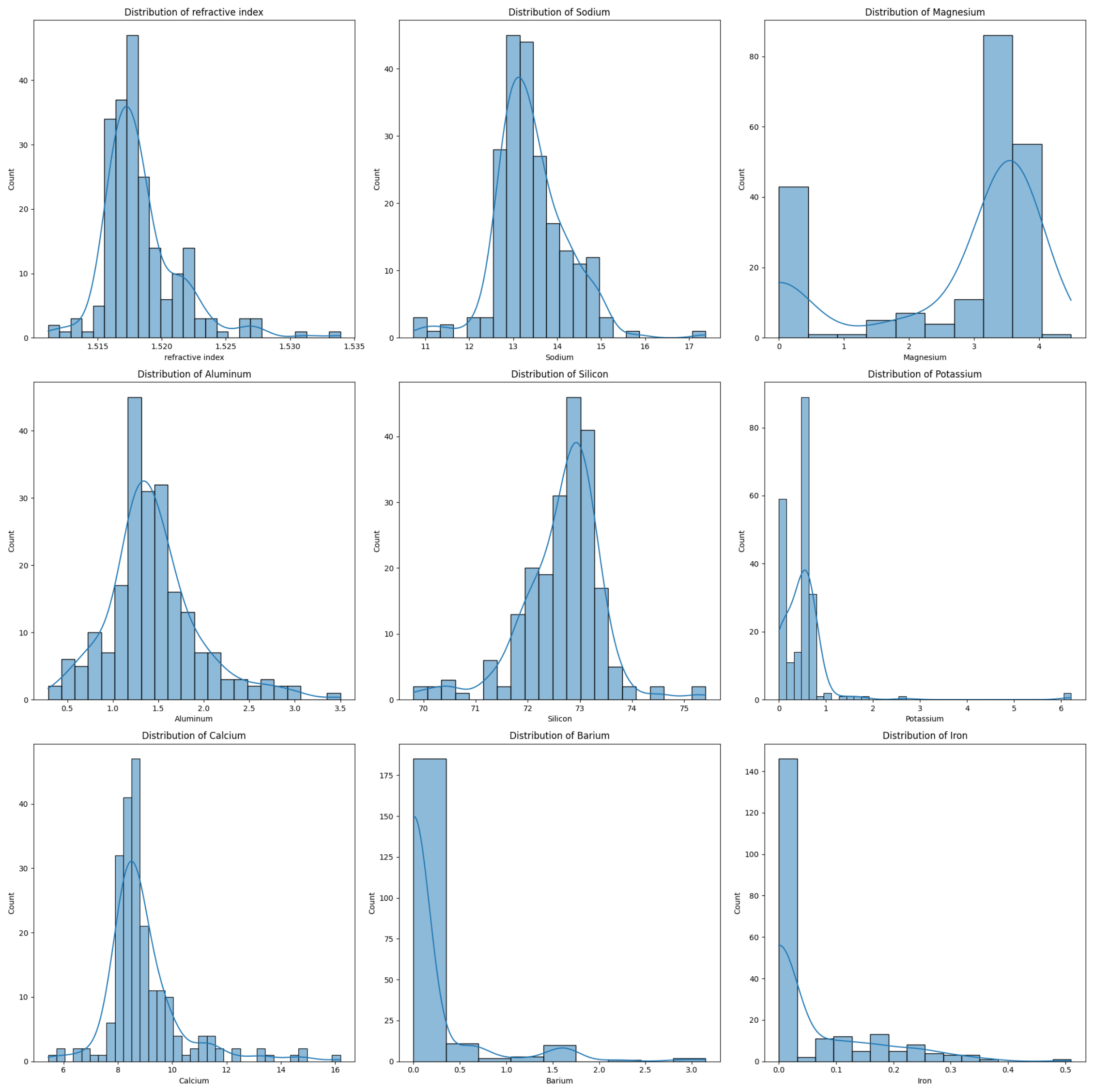
[4]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from ucimlrepo import fetch_ucirepo
import ssl
from sklearn.impute import SimpleImputer
import numpy as np
from scipy.stats import boxcox
from sklearn.preprocessing import PowerTransformer


ssl._create_default_https_context = ssl._create_unverified_context
glass_identification = fetch_ucirepo(name='Glass Identification')

X = glass_identification.data.features
```

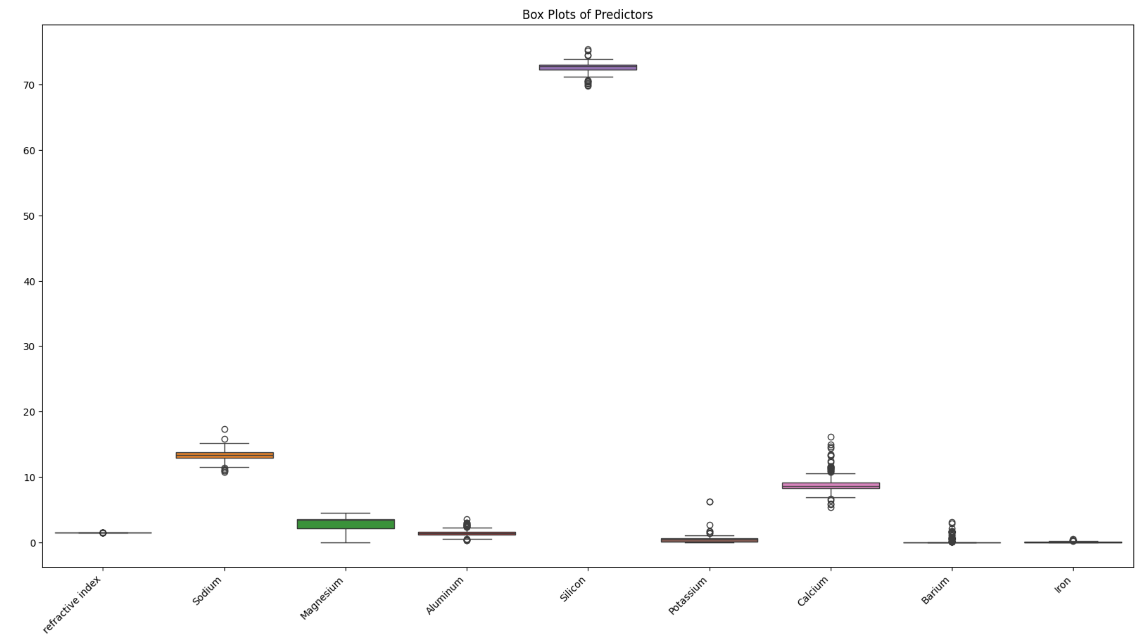
```
y = glass_identification.data.targets
```

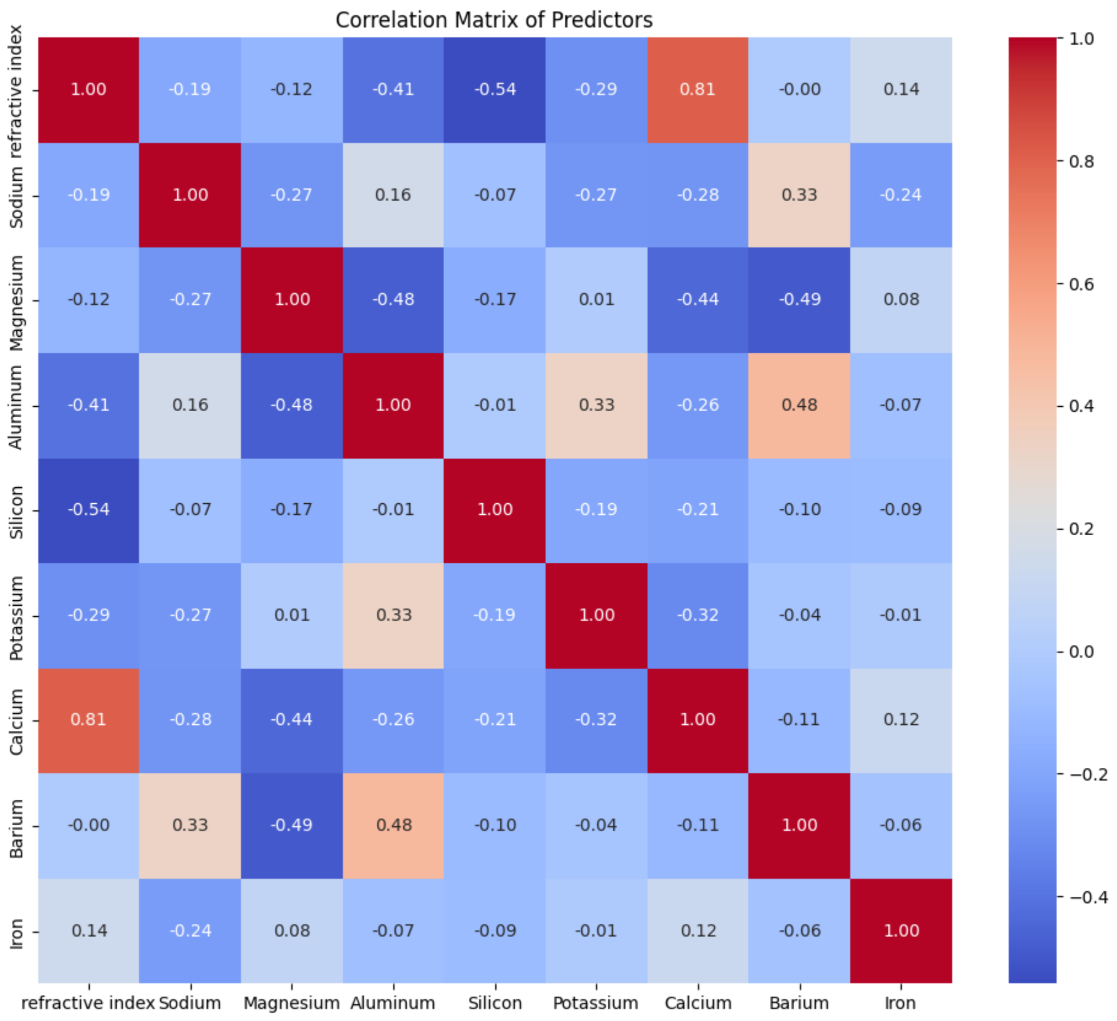
1.1.1 A. Exploratory data analysis through visualizations

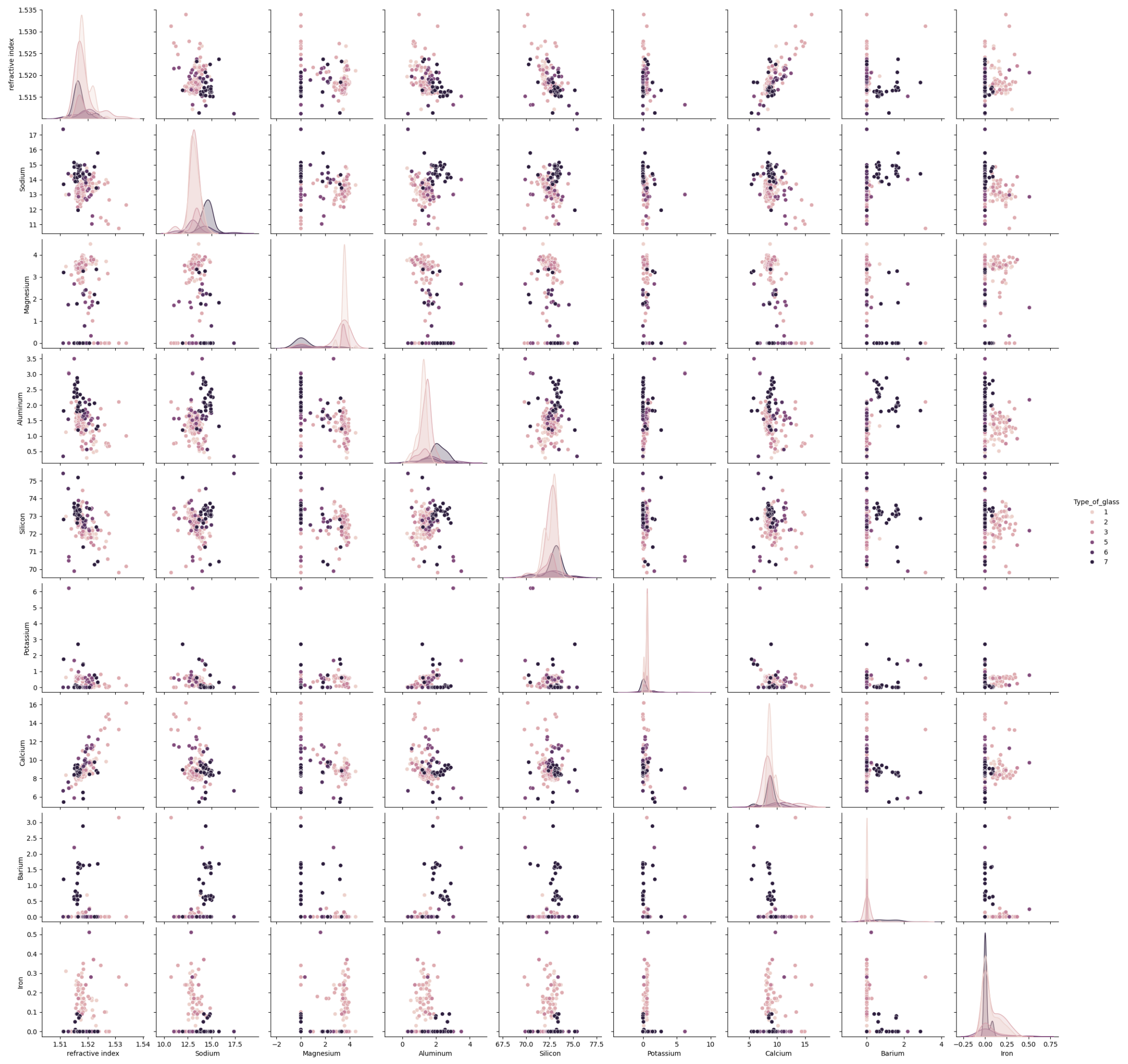
This first part shows the following by predictor: * Histograms * Box plots * Correlation matrix * Pairplots

[5]:
```
full_salts = {row['name']: row['description'] for _, row in␣
 ↪glass_identification.variables.iterrows() if row['name'] in X.columns}

# Rename the columns of X
X = X.rename(columns=full_salts)
Glass = pd.concat([X, y], axis=1)



#histograms
plt.figure(figsize=(20, 20))
for i, column in enumerate(X.columns, 1):
    plt.subplot(3, 3, i)
    sns.histplot(X[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
plt.tight_layout()
plt.show()

#boxplots
plt.figure(figsize=(20, 10))
sns.boxplot(data=X)
plt.xticks(rotation=45, ha='right')
plt.title('Box Plots of Predictors')
plt.show()

# correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(X.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Predictors')
plt.show()

# pairplots
sns.pairplot(Glass, hue=y.columns[0])
plt.show()
```


Distribution of refractive index
refractive index
Count
Distribution of Sodium
Sodium
Count
Distribution of Magnesium
Magnesium
Count
Distribution of Aluminum
Aluminum
Count
Distribution of Silicon
Silicon
Count
Distribution of Potassium
Potassium
Count
Distribution of Calcium
Calcium
Count
Distribution of Barium
Barium
Count
Distribution of Iron
Iron
Count


Box Plots of Predictors
0
10
20
30
40
50
60
70
refractive index
Sodium
Magnesium
Aluminum
Silicon
Potassium
Calcium
Barium
Iron


Correlation Matrix of Predictors
refractive index
Sodium
Magnesium
Aluminum
Silicon
Potassium
Calcium
Barium
Iron
1.00 -0.19 -0.12 -0.41 -0.54 -0.29 0.81 -0.00 0.14
-0.19 1.00 -0.27 0.16 -0.07 -0.27 -0.28 0.33 -0.24
-0.12 -0.27 1.00 -0.48 -0.17 0.01 -0.44 -0.49 0.08
-0.41 0.16 -0.48 1.00 -0.01 0.33 -0.26 0.48 -0.07
-0.54 -0.07 -0.17 -0.01 1.00 -0.19 -0.21 -0.10 -0.09
-0.29 -0.27 0.01 0.33 -0.19 1.00 -0.32 -0.04 -0.01
0.81 -0.28 -0.44 -0.26 -0.21 -0.32 1.00 -0.11 0.12
-0.00 0.33 -0.49 0.48 -0.10 -0.04 -0.11 1.00 -0.06
0.14 -0.24 0.08 -0.07 -0.09 -0.01 0.12 -0.06 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4



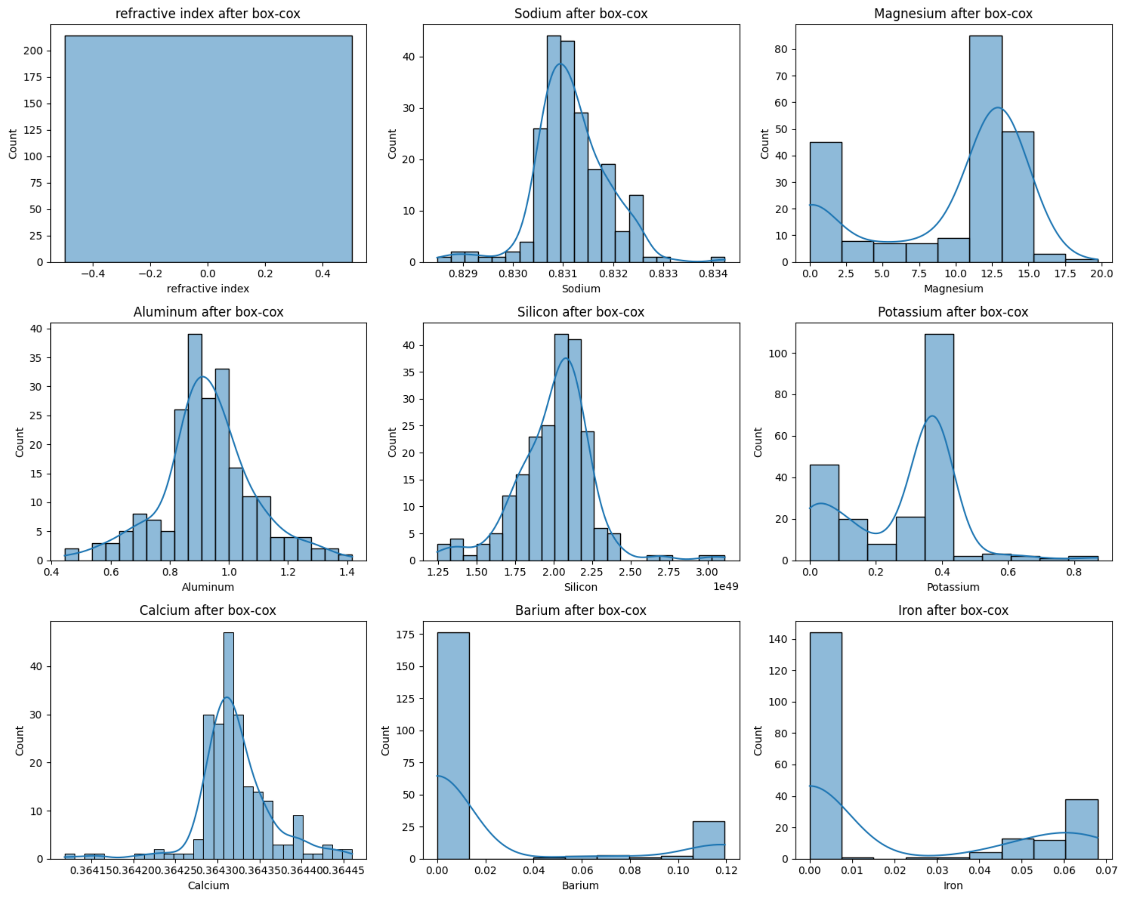
1.1.2 B. Do there appear to be any outliers in the data? Are any predictors skewed?

Answer: * Calcium and Refractivve Index have a strong negative relationship of -.81% * Potasium, Barium, and Iron have strong right skews * Silicon has the closest thing to a normal curve * Silicon is also causing heavy distortions in the box plot charts as its overall values are much higher than the others

1.1.3 C. Are there any relevant transformations of one or more predictors that might improve the classification model?

Answer: I went with a box-cox transformation since we've learned a lot about it thus far. I chose it because I wanted to try and normalize the curves a bit and see if I could reduce some of the variance. It had the intended effect, to an extent. Sodium was brough down much closer to where the others had previous been. Refractive Index was exploded in a very weird way, though. going from a normalish curive to just a full colored box.

[11]:
```python
X_transformed = X.copy()

if (X_transformed <= 0).any().any():
    X_positive = X_transformed + np.abs(X_transformed.min()) + 1
else:
    X_positive = X_transformed

pt = PowerTransformer(method='box-cox', standardize=False)

X_boxcox = pd.DataFrame(pt.fit_transform(X_positive), columns=X.columns)
Glass_transformed = pd.concat([X_boxcox, y], axis=1)

plt.figure(figsize=(15, 12))
for i, column in enumerate(X_boxcox.columns, 1):
    plt.subplot(3, 3, i)
    sns.histplot(X_boxcox[column], kde=True)
    plt.title(f"{column} after box-cox")

plt.tight_layout()
plt.show()
```



1.2 3.2 - UC Irvine Machine Learning Repository Soyboy Dataset Review

This exercise is a series of data cleaning questions about the Soybean(Large) dataset from the UCIML.

Per the exercise question from the book, "data were collected to predict disease in 683 soybeans. The 35 predictors are mostly categorical and include information on the environmental conditions (e.g., temperature, precipitation) and plant conditions (e.g., left spots, mold growth). The outcome labels consist of 19 distinct classes."

The questions are:

A. Investigate the frequency distributions for the categorical predictors. Are any of the distributions degenerate in the ways discussed earlier in this chapter?
B. Roughly 18% of the data are missing. Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?
C. Develop a strategy for handling missing data, either by eliminating predictors or imputation.

First, import and setup the data:

```
[10]: import math
      from matplotlib.ticker import PercentFormatter
      soybean_large = fetch_ucirepo(id=90)
      X = soybean_large.data.features
      y = soybean_large.data.targets
```

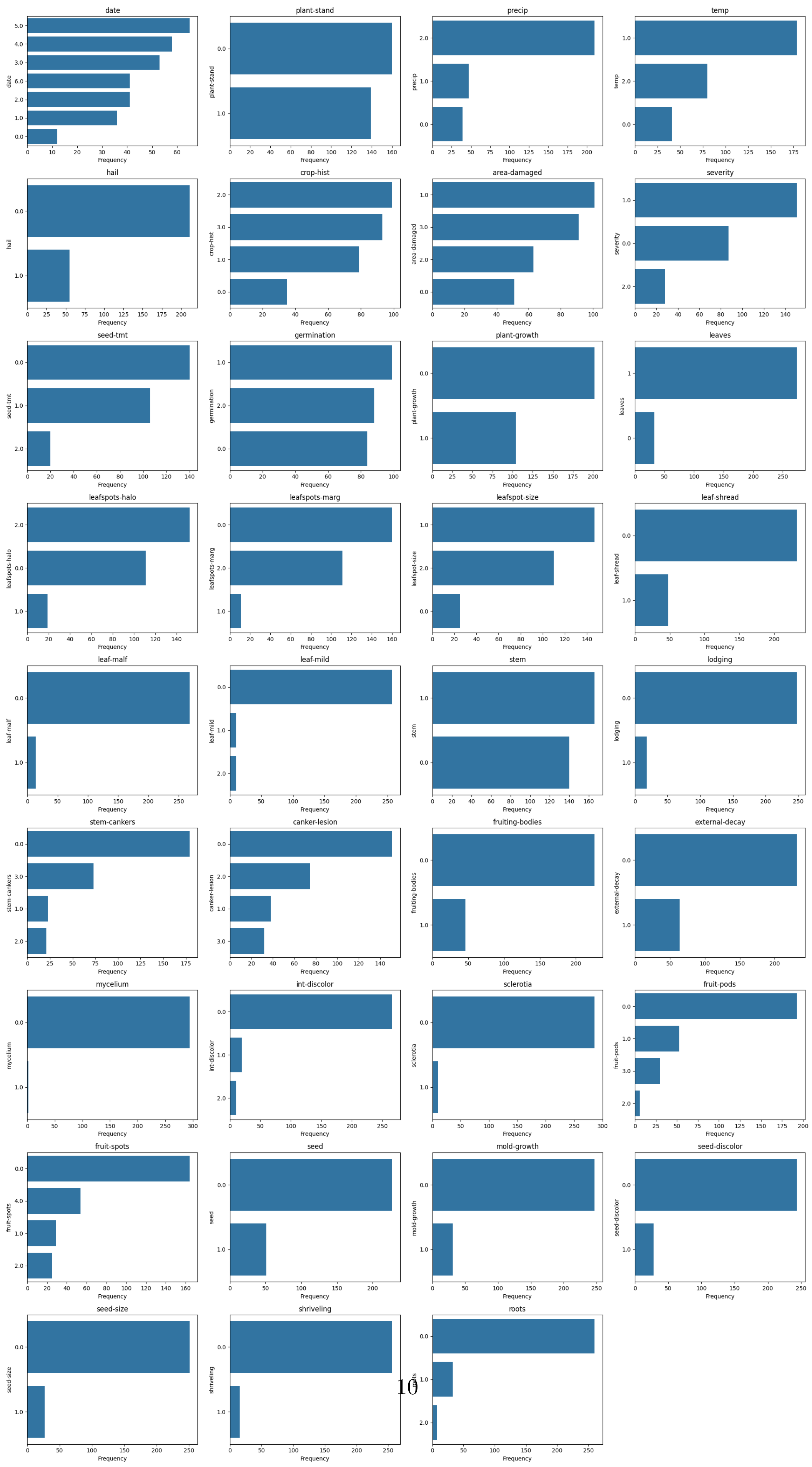
1.2.1 A. Investigate the frequency distributions for the categorical predictors

Are any of the distributions degenerate in the ways discussed earlier in this chapter?

Answer: More of these categorical predictors are degenerater than I would have initially guessed. For example, Leaf-Malf is very close to being a zero variance predictor, where almost all the values belong to 0 and hardly any to one. Mycelium and Sclerotia are in the same bucket. Compare this against Date, Crop-Hist, or Area-Damages each have healthy distributions across zero to three. This will make them easier to model since they have more data in different variety.

```
[12]: num_cols = 4
      num_rows = math.ceil(len(X.columns) / num_cols)
      fig, axes = plt.subplots(num_rows, num_cols, figsize=(20, num_rows * 4))
      axes = axes.flatten()
      for i, col in enumerate(X.columns):
          sns.countplot(y=X[col], order=X[col].value_counts().index, ax=axes[i],␣
       ↪hue=None, legend=False)
          axes[i].set_title(f'{col}')
          axes[i].set_xlabel('Frequency')


      for j in range(i + 1, num_rows * num_cols):
          fig.delaxes(axes[j])
```

```
plt.tight_layout()
plt.show()
```


date
plant-stand
precip
temp
hail
crop-hist
area-damaged
severity
seed-tmt
germination
plant-growth
leaves
leafspots-halo
leafspots-marg
leafspot-size
leaf-shread
leaf-malf
leaf-mild
stem
lodging
stem-cankers
canker-lesion
fruiting-bodies
external-decay
mycelium
int-discolor
sclerotia
fruit-pods
fruit-spots
seed
mold-growth
seed-discolor
seed-size
shriveling
roots
Frequency

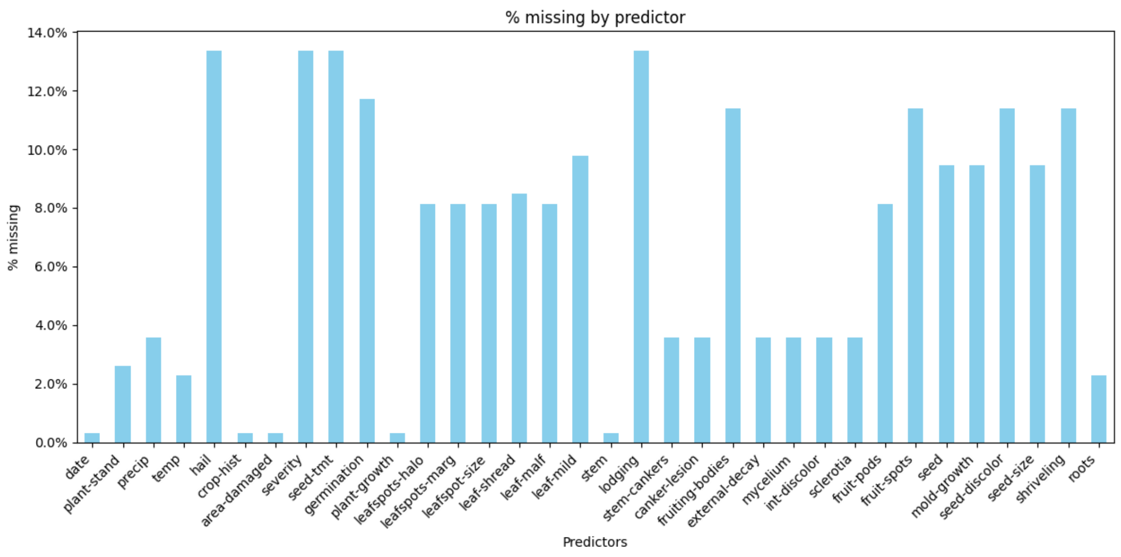
1.2.2 B. Investing missing 18% of data

Are there particular predictors that are more likely to be missing? Is the pattern of missing data related to the classes?

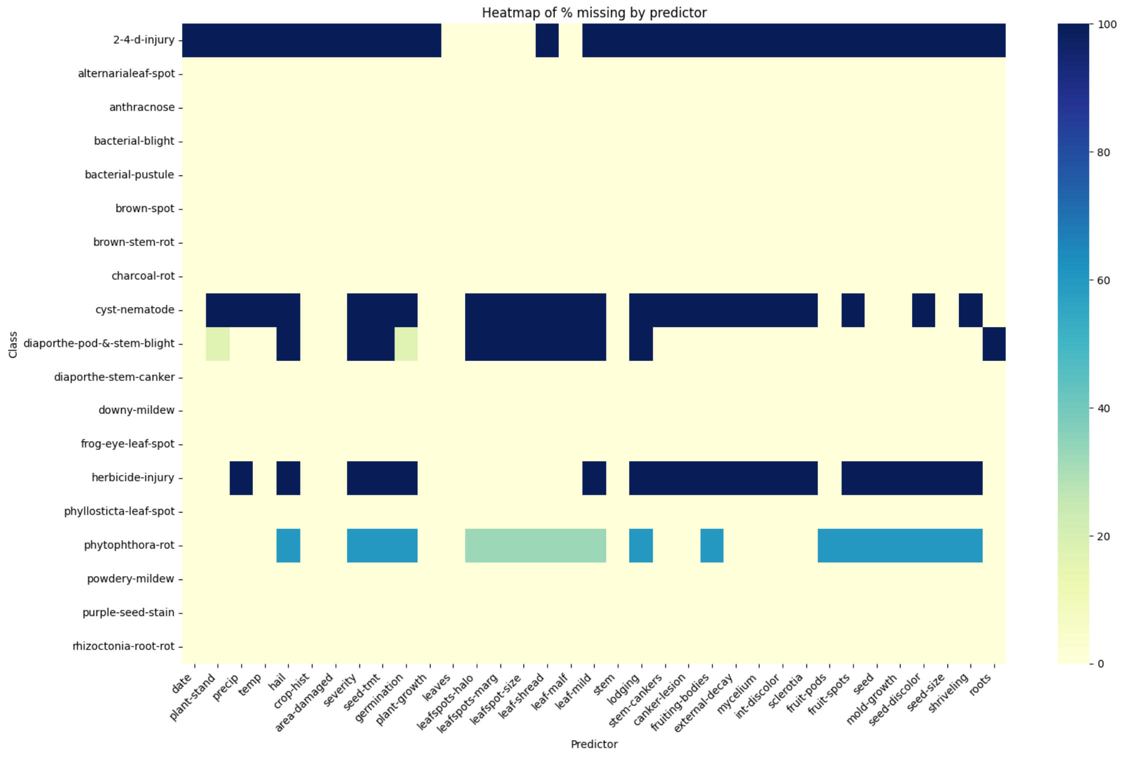
Answer: to answer this question, I made two charts, both normalized as % missing: * A basic bar chart showing the distribution of missing data across the entire datset, normalized as percent missing * A heatmap showing % missing by predictor class

It seems like the highest clustering is around some sort of damage to the seeds themselves, which could mean it's harder data to collect and organize effectively relative to others: * germination * fruit-spots * seed-discolor

[28]:
```python
soybean_large = fetch_ucirepo(id=90)
X = soybean_large.data.features
y = soybean_large.data.targets

missing_data = X.isnull().mean() * 100
md_filtered = missing_data[missing_data > 0]

plt.figure(figsize=(12, 6))
md_filtered.plot(kind='bar', color='skyblue')
plt.xlabel('Predictors')
plt.ylabel('% missing')
plt.title('% missing by predictor')
plt.xticks(rotation=45, ha='right')

plt.gca().yaxis.set_major_formatter(PercentFormatter())
plt.tight_layout()
plt.show()
```



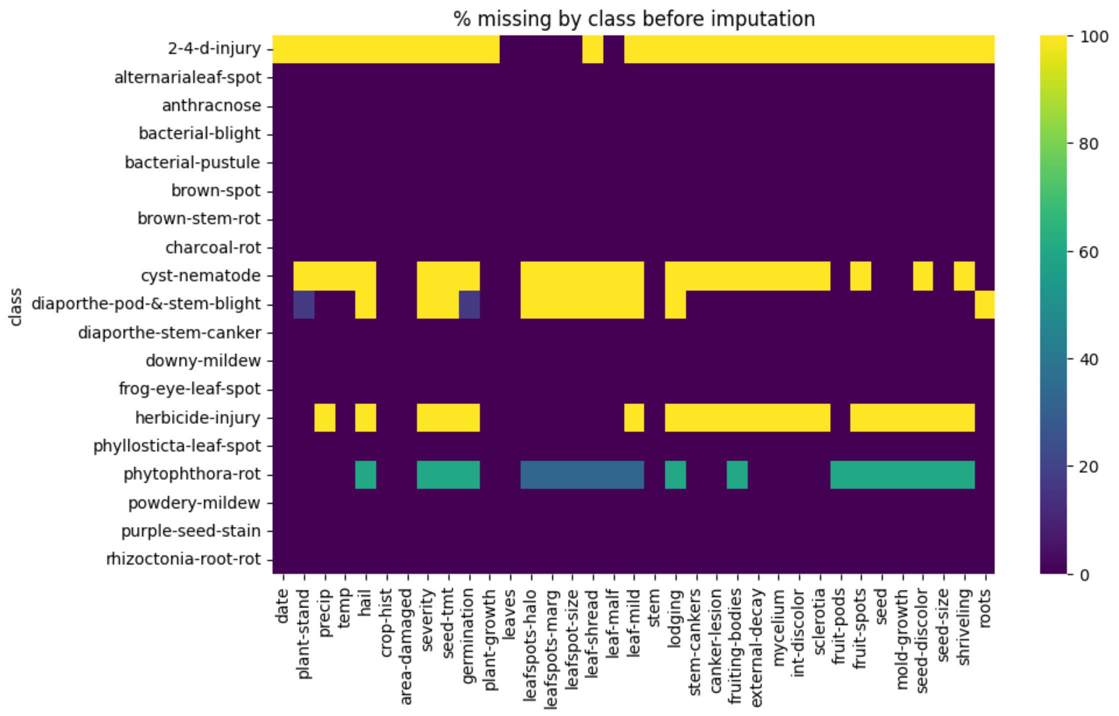
```
[39]: y = y.squeeze()
      missing_by_class = X.isnull().groupby(y).mean() * 100

      plt.figure(figsize=(16, 10))
      sns.heatmap(missing_by_class, cmap='YlGnBu', annot=False, fmt=".1f")
      plt.title('Heatmap of % missing by predictor')
      plt.ylabel('Class')
      plt.xlabel('Predictor')
      plt.xticks(rotation=45, ha='right')
      plt.tight_layout()
      plt.show()
```

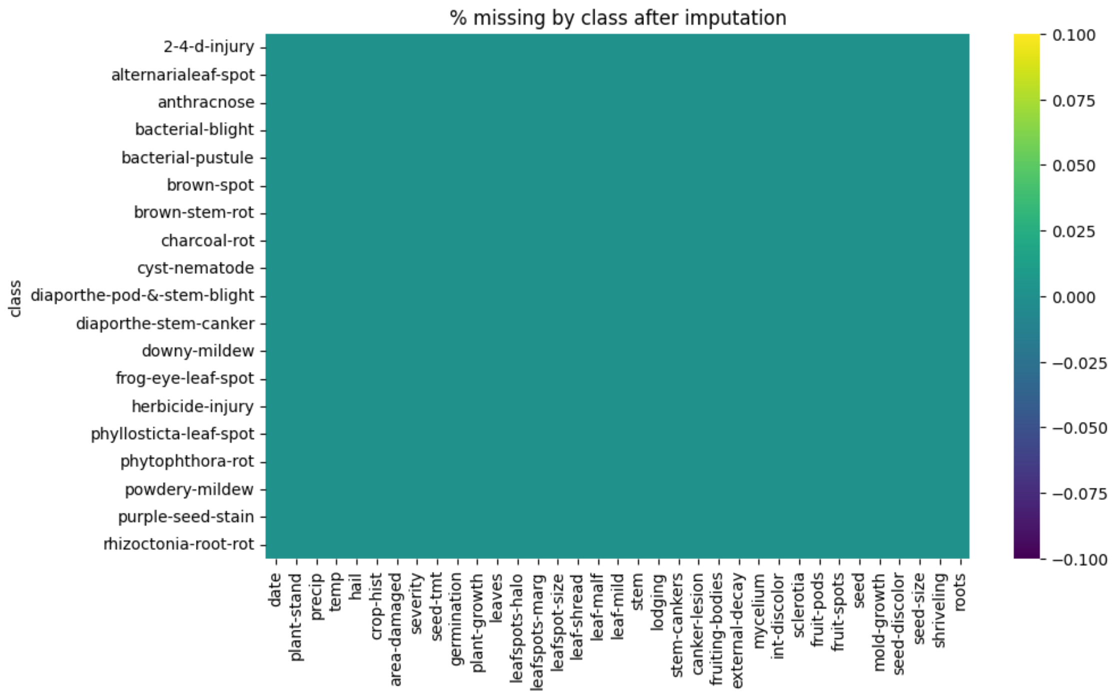


1.2.3 C. Strategy for handling missing data

Answer: I'm going to use a simple imputation strategy where the most frequent value is put in place of the missing values. I could do by median or mean as well but I was curious to see what this would do. This change is represented as before and after heatmaps below, showing the missing values being replaced.

[38]:
```
imputer = SimpleImputer(strategy='most_frequent')
X_imputed = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)
y = y.squeeze()

missing_before = X.isnull().groupby(y).mean() * 100
plt.figure(figsize=(10, 6))
sns.heatmap(missing_before, annot=False, cmap='viridis', cbar=True)
plt.title('% missing by class before imputation')
plt.show()


missing_after = X_imputed.isnull().groupby(y).mean() * 100
plt.figure(figsize=(10, 6))
sns.heatmap(missing_after, annot=False, cmap='viridis', cbar=True)
plt.title('% missing by class after imputation')
plt.show()
```




% missing by class after imputation
class
2-4-d-injury
alternarialeaf-spot
anthracnose
bacterial-blight
bacterial-pustule
brown-spot
brown-stem-rot
charcoal-rot
cyst-nematode
diaporthe-pod-&-stem-blight
diaporthe-stem-canker
downy-mildew
frog-eye-leaf-spot
herbicide-injury
phyllosticta-leaf-spot
phytophthora-rot
powdery-mildew
purple-seed-stain
rhizoctonia-root-rot
date
plant-stand
precip
temp
hail
crop-hist
area-damaged
severity
seed-tmt
germination
plant-growth
leaves
leafspots-halo
leafspots-marg
leafspot-size
leaf-shread
leaf-malf
leaf-mild
stem
lodging
stem-cankers
canker-lesion
fruiting-bodies
external-decay
mycelium
int-discolor
sclerotia
fruit-pods
fruit-spots
seed
mold-growth
seed-discolor
seed-size
shriveling
roots
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

```
[ ]:
```