CA3

March 21, 2023

1 DAT200 CA3 2023

Kaggle username: Jorid Holmen

1.0.1 Imports

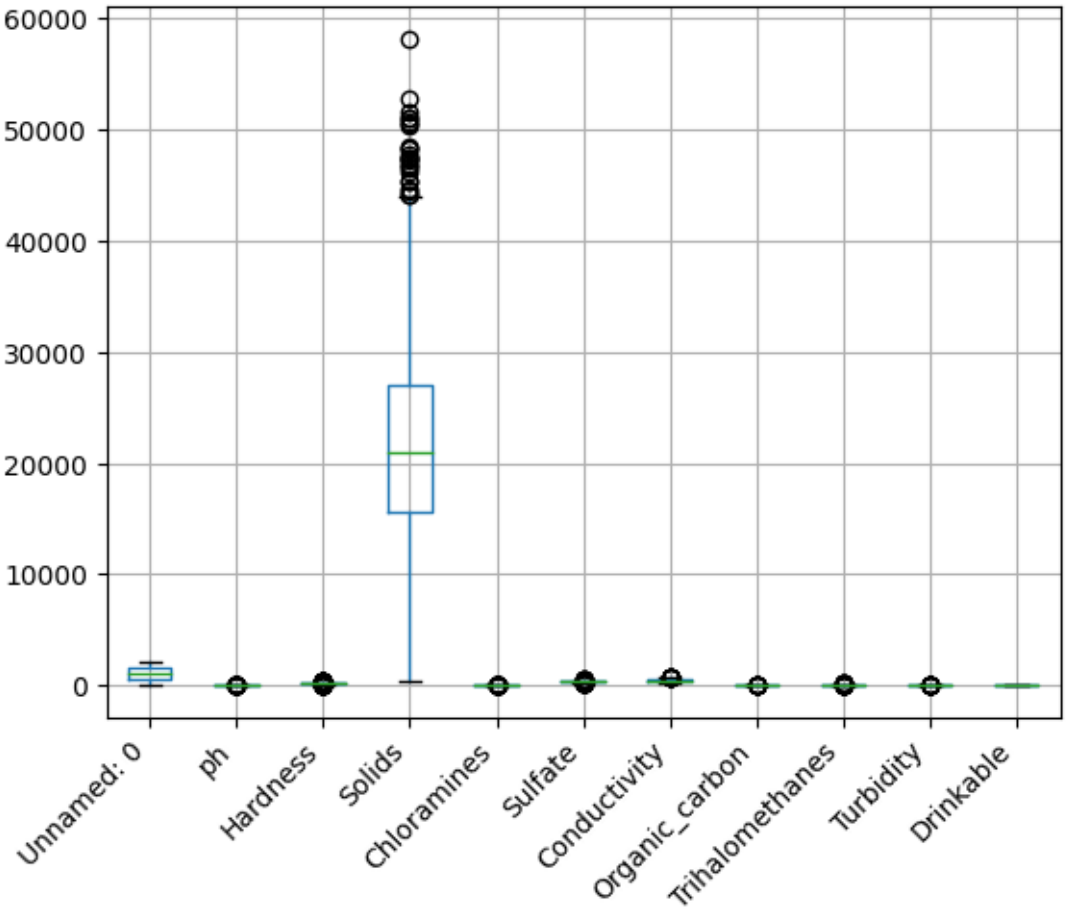
[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
from sklearn.metrics import accuracy_score
```

1.0.2 Reading data

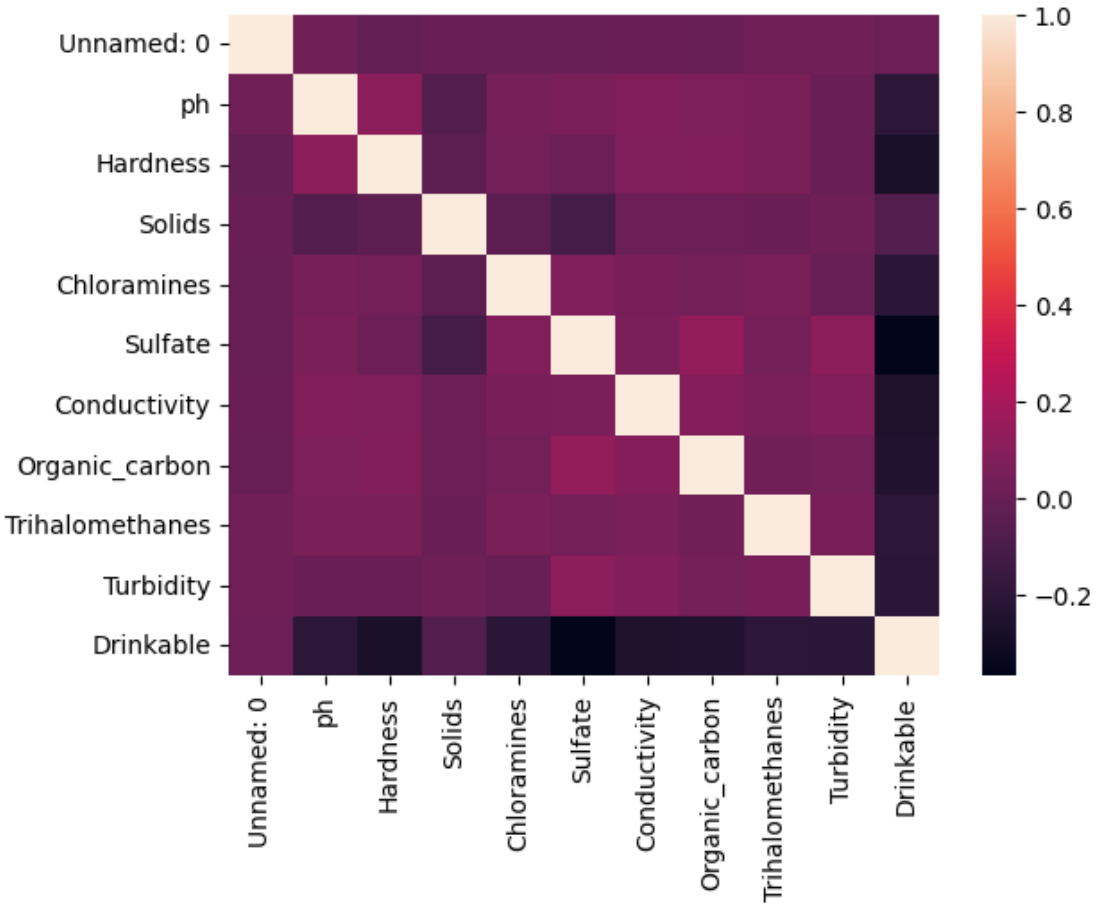
[ ]:
```
train = pd.read_csv('assets/train.csv')
test = pd.read_csv('assets/test.csv')
```

1.0.3 Data exploration and visualisation

[ ]:
```
train.boxplot()
plt.xticks(rotation=45, ha='right')
plt.show()
```

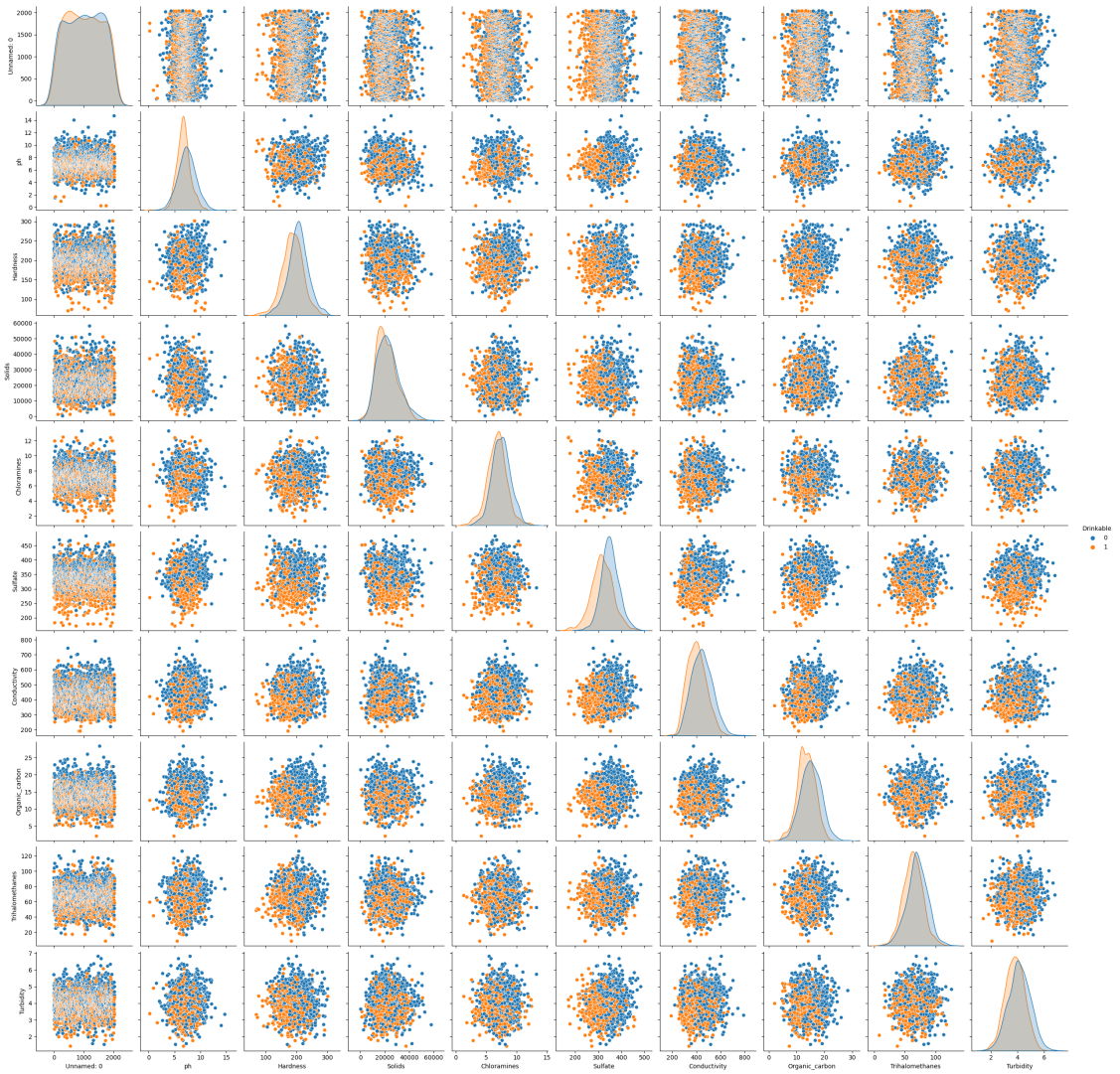


[ ]:
```
sns.heatmap(train.corr())
plt.show()
```



The plot shows low correlation between solids and sulfate.

[ ]:
```
sns.pairplot(train, hue='Drinkable')
plt.show()
```



1.0.4 Data cleaning

[ ]:
```
# checking for NaN values
print(f'Column Number of missing values ')
for c in train.columns:
    n_NaN = train[c].isnull().sum()
    print(f'{c:<32} {n_NaN}')
```

```
Column Number of missing values
Unnamed: 0                      0
ph                              0
Hardness                        0
Solids                          0
Chloramines                     0
```

```
Sulfate                 0
Conductivity            0
Organic_carbon          0
Trihalomethanes         0
Turbidity               0
Drinkable               0
```

There is no NaN values

[ ]:
```
# remove duplicates
train = train.drop_duplicates()
train.shape
```

[ ]:
```
(2040, 11)
```

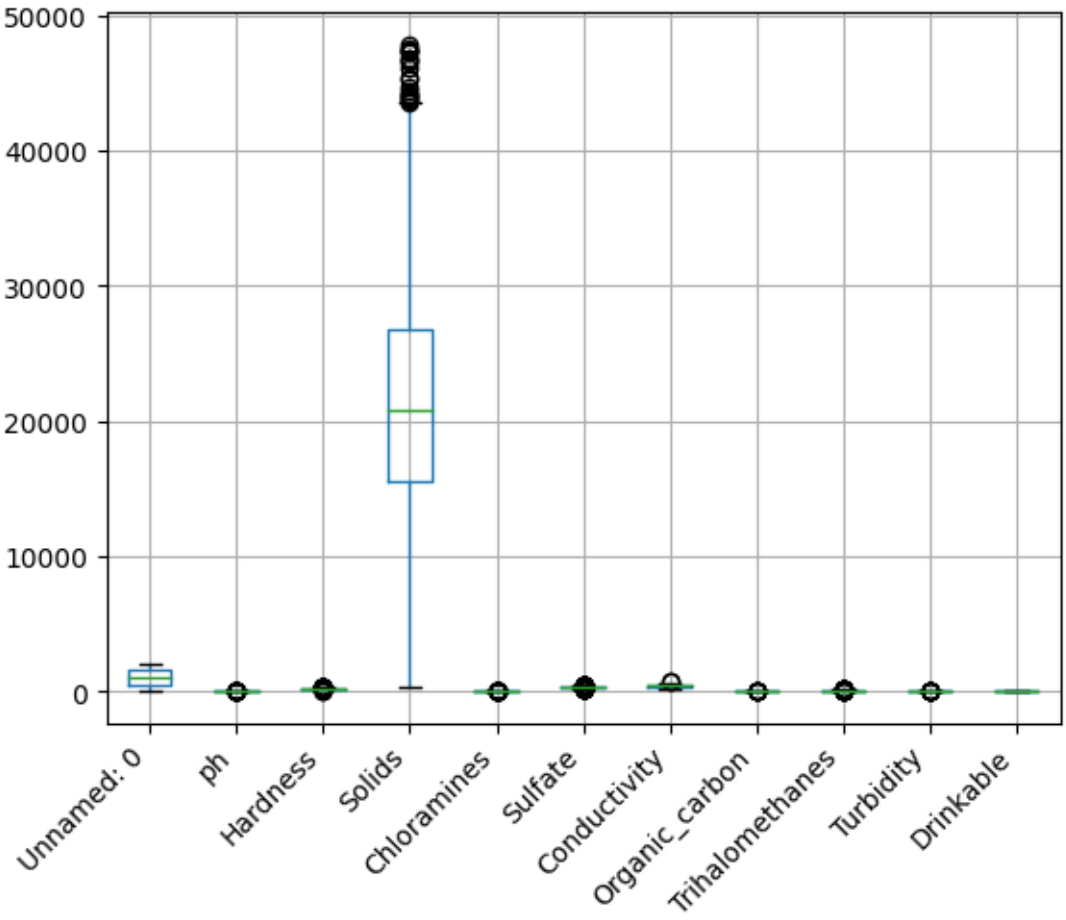
[ ]:
```
# remove outliers
z = np.abs(stats.zscore(train))
train = train[(z<3).all(axis=1)]

train.shape
```

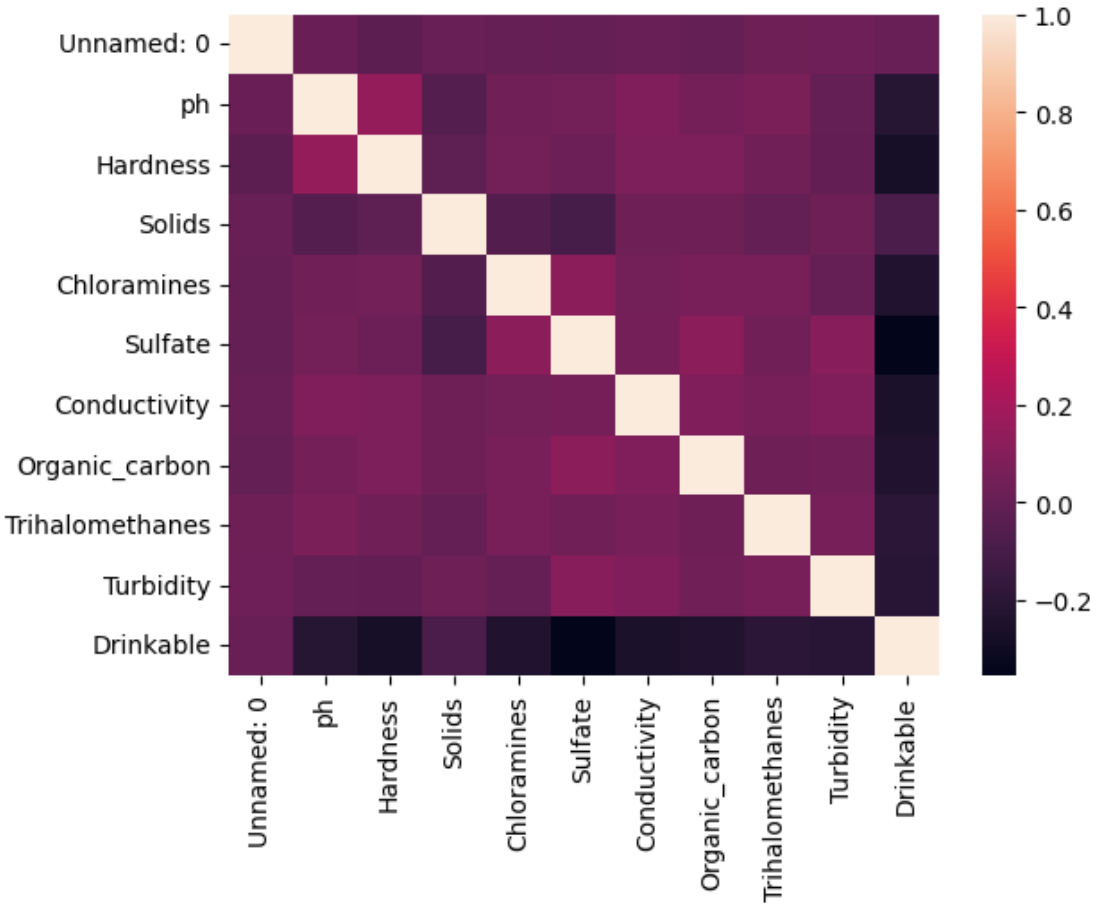
[ ]:
```
(1955, 11)
```

1.0.5 Data exploration after cleaning

[ ]:
```
# plot the boxplot to see the change the cleaning did
train.boxplot()
plt.xticks(rotation=45, ha='right')
plt.show()
```



[ ]:
```
sns.heatmap(train.corr())
plt.show()
```



1.0.6 Data preprocessing

[ ]:
```
# splitting into X and y
y = train.Drinkable
X = train.iloc[:,0:10].values
```

Train test split

[ ]:
```
# Split data into training and test data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=6, stratify=y)
```

Scaling Since deciding to use the Random forest classifier, there is no need for scaling.

1.0.7 Modelling

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

[ ]:
```
# Random forest
# Testing with  different n_estimators and random_state to see what works best

n_est = np.arange(100,1001,100)
r_state = np.arange(1,10)
l = []
for n_e in n_est:
    for r_s in r_state:
        rf = RandomForestClassifier(n_estimators=n_e,
                                    random_state=r_s)
        rf.fit(X_train, y_train)
        y_pred = rf.predict(X_test)
        l.append([accuracy_score(y_test, y_pred), f'r={r_s}', f'n={n_e}'])

maks = max(l)
print(maks) # printing out the best accuracy to know which combination works␣
 ↪best
```

```
[0.9011925042589438, 'r=4', 'n=100']
```

1.0.8 Final Evaluation

1.0.9 Kaggle submission

[ ]:
```
import csv

# n_estimator=100 and random_state=4 seems to work the best
rf = RandomForestClassifier(n_estimators=100,
                            random_state=4)

rf.fit(X, y)
y_pred = rf.predict(test.values)

# write the results to a csv file
with open('kaggle_submission.csv', 'w') as f:
    w = csv.writer(f)

    w.writerow(['index','Drinkable'])

    for r in range(0,360):
        w.writerow([r, y_pred[r]])
```