DAT200 CA5 2022

Kaggle username: jorid holmen

Imports

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.impute import SimpleImputer
import csv
from sklearn.metrics import r2_score

from sklearn.ensemble import RandomForestClassifier, RandomForestRegresso
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
```

Reading data

In [ ]:
```
train = pd.read_csv('data/train.csv', index_col=0)
test = pd.read_csv('data/test.csv', index_col=0)
```

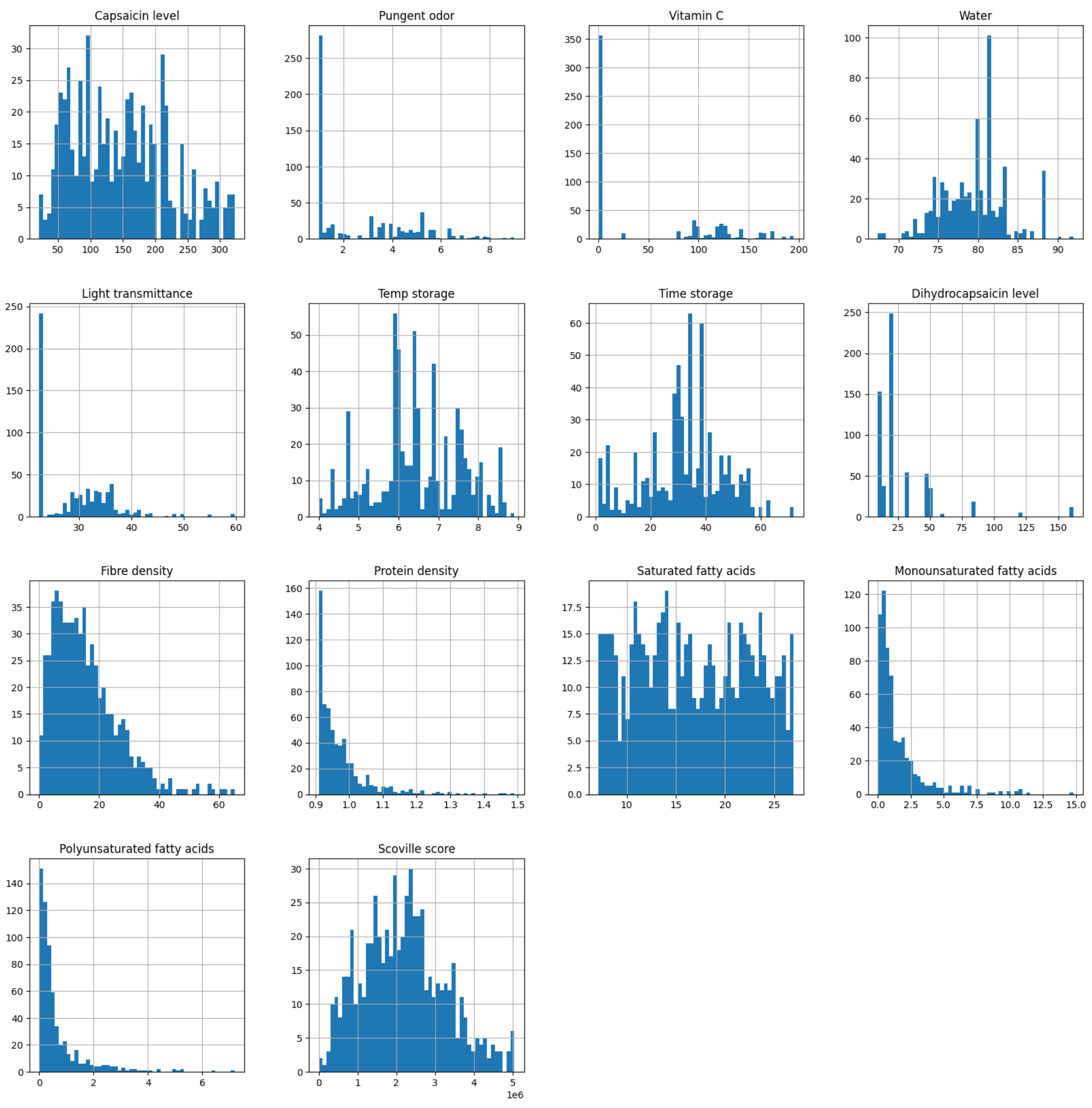
Data exploration and visualisation

In [ ]:
```
train.head()
```

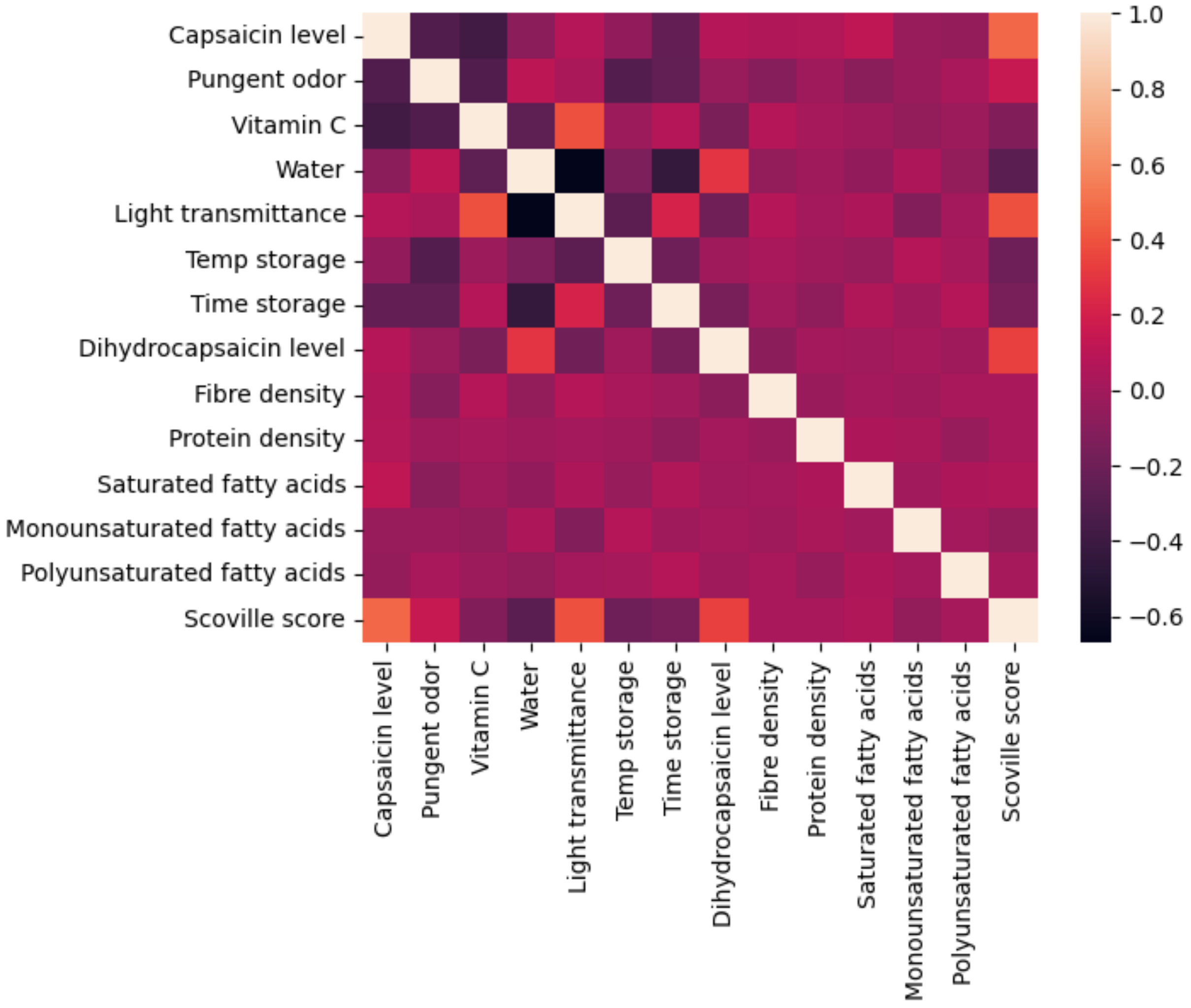
Out[ ]:

	Capsaicin level	Pungent odor	Vitamin C	Water	Light transmittance	Temp storage	Time storage	Dihydrocapsaici leve
0	166.7	6.8	0.0	77.95	32.4	7.5	4.0	32.
1	170.4	5.7	0.0	74.06	39.0	4.7	52.0	47
2	212.5	3.1	0.0	88.27	22.4	5.9	1.0	161.
3	216.0	1.0	0.0	80.01	22.4	6.4	31.0	10.
4	146.5	1.3	141.0	78.64	28.8	4.6	38.0	20.

```
In [ ]: train.hist(figsize=(20,20), bins=50)
        plt.show()
```



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



Data cleaning

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

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

```
Column Number of missing values in train data
Capsaicin level                  0
Pungent odor                     0
Vitamin C                        0
Water                            0
Light transmittance              0
Temp storage                     0
Time storage                     0
Dihydrocapsaicin level           0
Fibre density                    0
Protein density                  0
Saturated fatty acids            0
Monounsaturated fatty acids      0
Polyunsaturated fatty acids      0
Scoville score                   0
Column Number of missing values in test data
Capsaicin level                  0
Pungent odor                     0
Vitamin C                        0
Water                            0
Light transmittance              0
Temp storage                     0
Time storage                     0
Dihydrocapsaicin level           0
Fibre density                    0
Protein density                  0
Saturated fatty acids            0
Monounsaturated fatty acids      0
Polyunsaturated fatty acids      0
```

In [ ]:
```
# checking for outliers
# fjerne kategorier som har lav korrelasjon med scoville
```

Data exploration after cleaning

In [ ]:
```
```

Data preprocessing

In [ ]:
```
```

Train test split

In [ ]:
```
# splitting into X and y
y = train['Scoville score']
X = train.iloc[:,0:13]

# 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=1)
```

Scaling

```
In [ ]: # scaling in pipeline
```

Modelling

Data pipeline with regression model

```
In [ ]: pipe_rfreg = make_pipeline(RandomForestRegressor())

        param_grid = {'randomforestregressor__random_state': list(np.arange(100,1
                      'randomforestregressor__n_estimators': list(np.arange(1,10)

        gs_lr = GridSearchCV(estimator=pipe_rfreg,
                             param_grid=param_grid,
                             scoring='neg_mean_squared_error',
                             cv=10,
                             n_jobs=-1)

        gs_lr_test = gs_lr.fit(X_train, y_train)
        clf_rfr_best = gs_lr_test.best_estimator_
        clf_rfr_best.fit(X_train, y_train)
        print(r2_score(y_test, clf_rfr_best.predict(X_test)))
        print(gs_lr_test.best_estimator_)
```

```
0.7921992956541402
Pipeline(steps=[('randomforestregressor',
                 RandomForestRegressor(n_estimators=9, random_state=300)
)])
```

```
In [ ]: r = RandomForestRegressor(n_estimators=9, random_state=300)
        r.fit(X_train, y_train)
        print(r2_score(y_test, clf_rfr_best.predict(X_test)))
```

```
0.7921992956541402
```

```
In [ ]: clf_rfr_best.fit(X, y)
        y_pred = clf_rfr_best.predict(test)

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

            w.writerow(['Id','Scoville score'])

            for r in range(0, 412):
                w.writerow([r, int(y_pred[r])])
```

Data pipeline with classification model

In [ ]:
```
y_train_binned = pd.cut(y_train, 10, labels=False)
y_test_binned = pd.cut(y_test, 10, labels=False)

pipe_ada = make_pipeline(StandardScaler(),
                         AdaBoostClassifier())

param_grid = {'adaboostclassifier__estimator': [DecisionTreeClassifier()]
              'adaboostclassifier__n_estimators': list(np.arange(1,10)),
              'adaboostclassifier__learning_rate': [0.05, 0.1, 0.15, 0.2]
              'adaboostclassifier__random_state': list(np.arange(100,1001

gs_ada = GridSearchCV(estimator=pipe_ada,
                  param_grid=param_grid,
                  scoring='neg_mean_squared_error',
                  cv=10,
                  n_jobs=-1)

gs_ada_test = gs_ada.fit(X_train, y_train_binned)
clf_ada_best = gs_ada_test.best_estimator_
clf_ada_best.fit(X_train, y_train_binned)
print(clf_ada_best.score(X_test, y_test_binned))
print(gs_ada_test.best_estimator_)
```

```
0.3602150537634409
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('adaboostclassifier',
                 AdaBoostClassifier(estimator=DecisionTreeClassifier(),
                                    learning_rate=0.05, n_estimators=1,
                                    random_state=800))])
```

In [ ]:
```
clf_ada_best.fit(X, y)
y_pred = clf_ada_best.predict(test)

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

    w.writerow(['Id','Scoville score'])

    for r in range(0, 412):
        w.writerow([r, int(y_pred[r])])
```

Other models used for Kaggle submission

In [ ]:

Final Evaluation

In [ ]:

Kaggle submission

```
In [ ]: r = RandomForestRegressor(n_estimators=9, random_state=300)
        r.fit(X, y)
        y_pred = r.predict(test)

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

            w.writerow(['Id','Scoville score'])

            for r in range(0, 412):
                w.writerow([r, int(y_pred[r])])
```