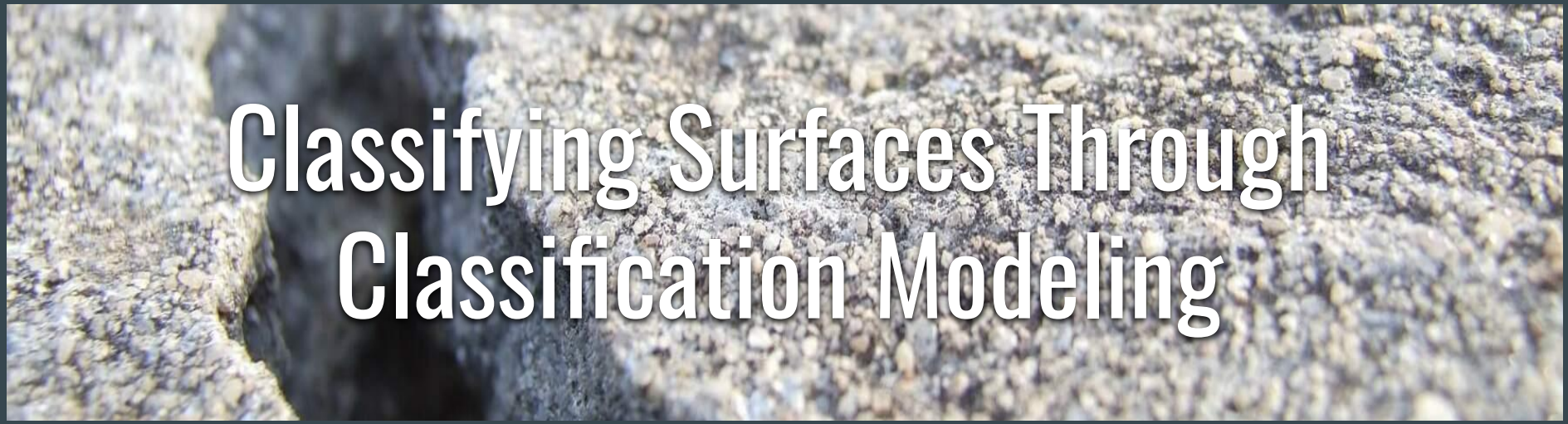


Classifying Surfaces Through Classification Modeling

Tyler Lehman

CMSE 381 Section_002 Final Project

Background & Motivation

- Classification & Regression
- Default Dataset
- Classification correctly cracked surfaces vs non-cracked?
- Testing a Support Vector Classifier (SVC) and a Decision Tree model
- Improvements through feature selection or cross-validation tuning?

Data & Imports

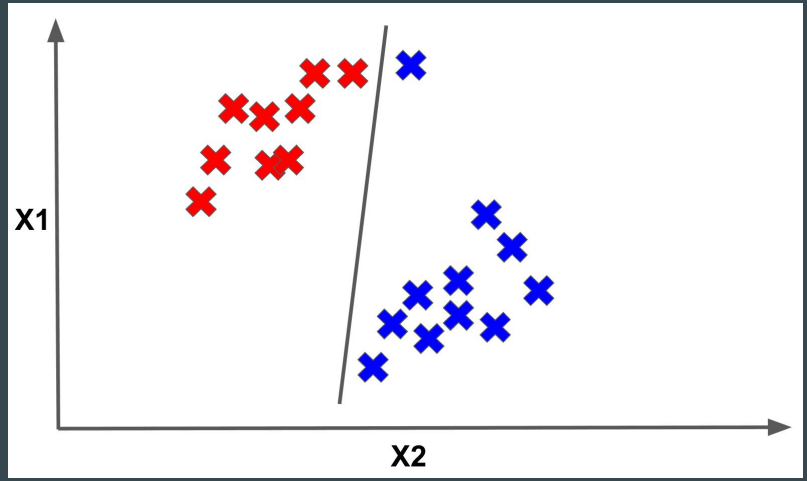
- Over 56,000 images
- Cracked vs Non-cracked
- Walls, Pavements, and Decks
- Common Classification Imports

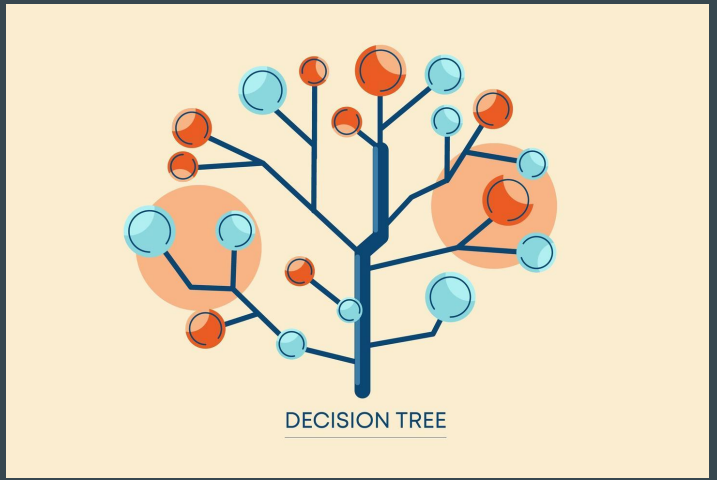
kaggle

```
# Imports for loading in the images
import os
from skimage.io import imread
from skimage.transform import resize

# SVC Imports
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Decision Tree Imports
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
```

Models for Classification

- Support Vector Classifier (SVC)
- Decision Tree

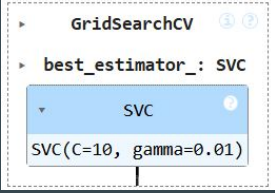




Methodology - SVC

```
# Code to find the folder that is the Default dataset. It's called archive (5) on my machine.
base_dir = 'C:/Users/pgleh/Downloads/archive (5)'

# Creating categories and labels that my model will use and classify images into.
categories = ['Walls', 'Pavements', 'Decks']
labels = ['Cracked', 'Non-cracked']

# data will store the pixel values and target will store corresponding labels for each image.
data = []
target = []
```

```
svc = SVC()

# Creating an amount of parameters to test and go through. Again, for
params = [{'gamma': [0.01, 0.001, 0.0001], 'C': [1, 10, 100, 1000]}]
# This trains 12 image classifiers, going through gamma and C params.

# Cross-validation technique for my SVC.
grid_search = GridSearchCV(svc, params, n_jobs = -1)
# Fitting the training data to the model.
grid_search.fit(x_train, y_train)
```




```
# I found Pipeline helps make feature selection run faster when I l
# This code snippet helps run through the code faster and will sele
pipeline = Pipeline([
    ('select', SelectKBest(score_func = f_classif)),  # Don't set k
    ('svc', SVC())])

params = {
    'select__k': [100, 300, 500, 1000, 2000],
    'svc__C': [1, 10, 1000, 1000],
    'svc__gamma': [0.0001, 0.001, 0.01]
}

# Fitting the model now to see what parameters are best.
grid_search = GridSearchCV(pipeline, params, cv = 5, n_jobs = -1)
grid_search.fit(x_train, y_train)

print("Best params:", grid_search.best_params_)
print("Best accuracy:", grid_search.best_score_)
```

Methodology - Decision Tree

```
tree = DecisionTreeClassifier(random_state = 42)
tree.fit(x_train, y_train)
```

DecisionTreeClassifier

DecisionTreeClassifier(random_state=42)

```
from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(tree, x_train, y_train, cv = 5)
```

0.56

	precision	recall	f1-score	support
Decks_Cracked	0.09	0.09	0.09	23
Decks_Non-cracked	0.48	0.61	0.54	92
Pavements_Cracked	0.14	0.15	0.14	20
Pavements_Non-cracked	0.74	0.75	0.75	183
Walls_Cracked	0.27	0.24	0.25	38
Walls_Non-cracked	0.60	0.51	0.55	144
accuracy			0.56	500
macro avg	0.39	0.39	0.39	500
weighted avg	0.56	0.56	0.56	500

Results

- Not too accurate
- SVC better performer
- K-Fold didn't help

score
0.68

0.56

	precision	recall	f1-score	support
Decks_Cracked	0.09	0.09	0.09	23
Decks_Non-cracked	0.48	0.61	0.54	92
Pavements_Cracked	0.14	0.15	0.14	20
Pavements_Non-cracked	0.74	0.75	0.75	183
Walls_Cracked	0.27	0.24	0.25	38
Walls_Non-cracked	0.60	0.51	0.55	144
accuracy			0.56	500
macro avg	0.39	0.39	0.39	500
weighted avg	0.56	0.56	0.56	500

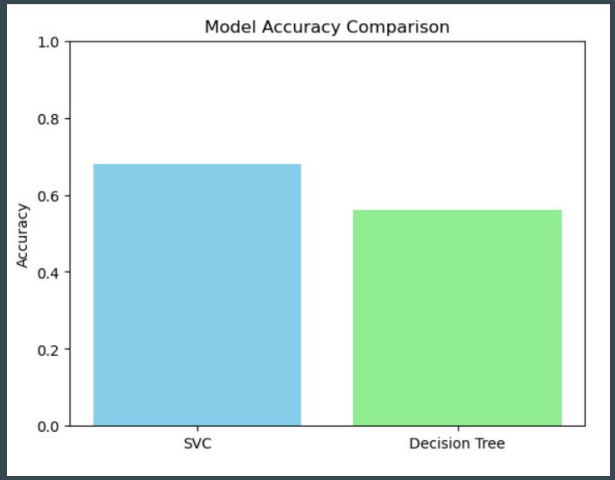
```
print("Best params:", grid_search.best_params_)
print("Best accuracy:", grid_search.best_score_)
```

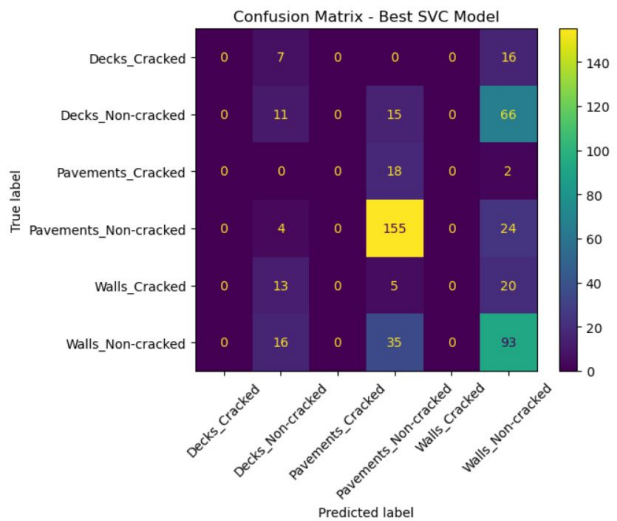
```
Best params: {'select__k': 2000, 'svc__C': 1, 'svc__gamma': 0.01}
Best accuracy: 0.54
```

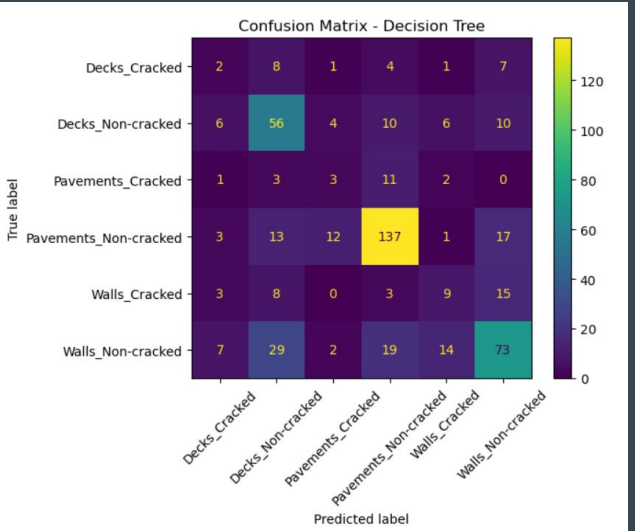
```
# k values were: [100, 300, 500, 1000, 2000]
print("Scores per Fold:", cv_scores)
print("Mean CV accuracy:", np.mean(cv_scores))
```

```
Scores per Fold: [0.52   0.5675 0.515  0.535  0.5425]
Mean CV accuracy: 0.536
```

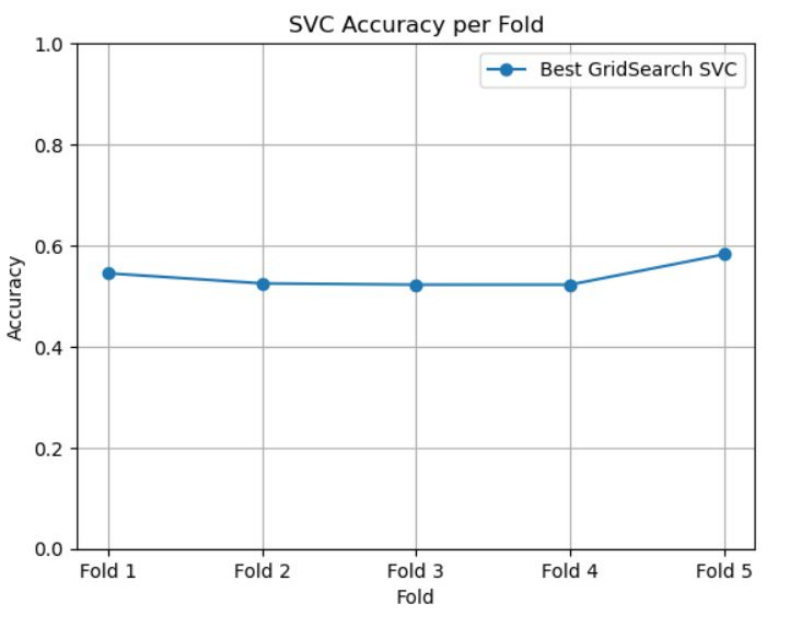
Results

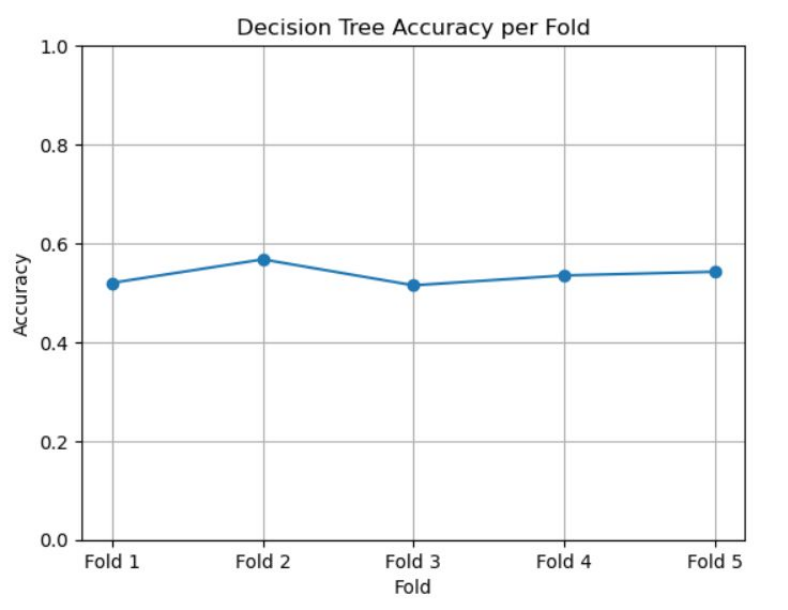

Model Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
SVC
Decision Tree


Confusion Matrix - Best SVC Model
Decks_Cracked
Decks_Non-cracked
Pavements_Cracked
Pavements_Non-cracked
Walls_Cracked
Walls_Non-cracked
True label
Predicted label
0 7 0 0 0 16
0 11 0 15 0 66
0 0 0 18 0 2
0 4 0 155 0 24
0 13 0 5 0 20
0 16 0 35 0 93
140
120
100
80
60
40
20
0


Confusion Matrix - Decision Tree
Decks_Cracked
Decks_Non-cracked
Pavements_Cracked
Pavements_Non-cracked
Walls_Cracked
Walls_Non-cracked
True label
Predicted label
2 8 1 4 1 7
6 56 4 10 6 10
1 3 3 11 2 0
3 13 12 137 1 17
3 8 0 3 9 15
7 29 2 19 14 73
120
100
80
60
40
20
0

Results


SVC Accuracy per Fold
Best GridSearch SVC
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold


Decision Tree Accuracy per Fold
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold

Conclusion

- Models do work, not accurate enough
- SVC better predictor
- K-Fold didn’t help much
- Next time:
- Use more data
- Try regression?

References

"8.3. Parallelism, Resource Management, and Configuration." Scikit, scikit-learn.org/stable/computing/parallelism.html. Accessed 25 Apr. 2025.

Chatgpt | Openai, openai.com/index/chatgpt/. Accessed 25 Apr. 2025.

Computer vision engineer. "Image Classification with Python and Scikit Learn | Computer Vision Tutorial." YouTube, YouTube, www.youtube.com/watch?v=il8dMDlXrIE. Accessed 25 Apr. 2025.

"Pipeline." Scikit, scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html. Accessed 25 Apr. 2025.

Whoami-As. "Structural Defects Network (SDNET) 2018." Kaggle, 28 July 2020, www.kaggle.com/datasets/aniruddhsharma/structural-defects-network-concrete-crack-images/data.

- I also used multiple of the inclass assignments for reference, including K-Fold CV and Classification lessons, Decision Trees, and SVC lessons.