Feature Extraction from Satellite Images

BTech End Semester Project Report

Phase 1

Lalit Brahma

Roll No. 200104050

BTech Supervisor: **Dr. Rishikesh Bharti**

CERTIFICATE

It is certified that the work contained in the project report entitled "**Feature Extraction from Satellite Images**", by **Lalit Brahma** (200104050) has been carried out under my/our supervision and that this work has not been submitted elsewhere for the award of a degree or diploma.

Date:

Signature

Dr. Rishikesh Bharti

Department of Civil Engineering

Indian Institute of Technology Guwahati

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to *Dr. Rishikesh Bharti* from the Civil Engineering department at the Indian Institute of Technology Guwahati for his invaluable contributions, dedicated time, and unwavering efforts in supporting the successful completion of our project, titled "***Feature Extraction from Satellite Images***". Dr. Rishikesh's insightful advice and constructive suggestions were pivotal in guiding me through the challenges encountered during the project, for which we are eternally grateful.

I am also deeply thankful to my supervisor, *Vatsal Patel*, for their continuous encouragement, patience, and mentorship. Their guidance has been pivotal in helping me navigate through the challenges of the research process and has significantly contributed to the quality of this report.

Furthermore, we express our sincere appreciation to the Civil Department at IIT Guwahati for providing me with the opportunity to engage in this impactful project and enhance our skills. The collaborative environment fostered by the department significantly contributed to our growth and learning throughout this endeavor.

Date: Signature of the student

Lalit Brahma

INDEX

INTRODUCTION

Pavement distress conditions, which encompass a wide range of structural and functional issues in road surfaces, are a critical aspect of transportation infrastructure management. Assessing these distress conditions is vital for maintaining safe, efficient, and cost-effective road networks. Pavement distresses can manifest in various forms, including cracks, potholes, rutting, roughness, and surface deformations. These distresses result from a combination of factors, such as traffic loads, environmental conditions, material quality, and construction practices.

Different types of pavement distress

Assessment of pavement distress nowadays encompasses several types and technologies:

- ***Visual Inspection***: A fundamental part of pavement assessment involves trained inspectors visually examining the road surface. They look for cracks, surface wear, potholes, and other observable distresses. This on-site assessment provides essential information about the extent and severity of the problems.
- ***Data Collection***: Pavement distress data is collected through various means, such as mobile data collection units equipped with sensors and cameras. These systems capture high-resolution images and measurements of the road surface, including the type and extent of distress.
- ***Non-Destructive Testing***: Advanced non-destructive testing techniques, such as Ground Penetrating Radar (GPR) and Falling Weight Deflectometer (FWD), are used to assess the structural integrity of the pavement layers and subgrade. These tests provide valuable insights into the overall condition and load-bearing capacity of the pavement.
- ***Geographic Information Systems (GIS):*** GIS technology is utilized to create comprehensive databases of pavement distress data, enabling transportation agencies to prioritize maintenance and rehabilitation projects based on the severity and location of distress.

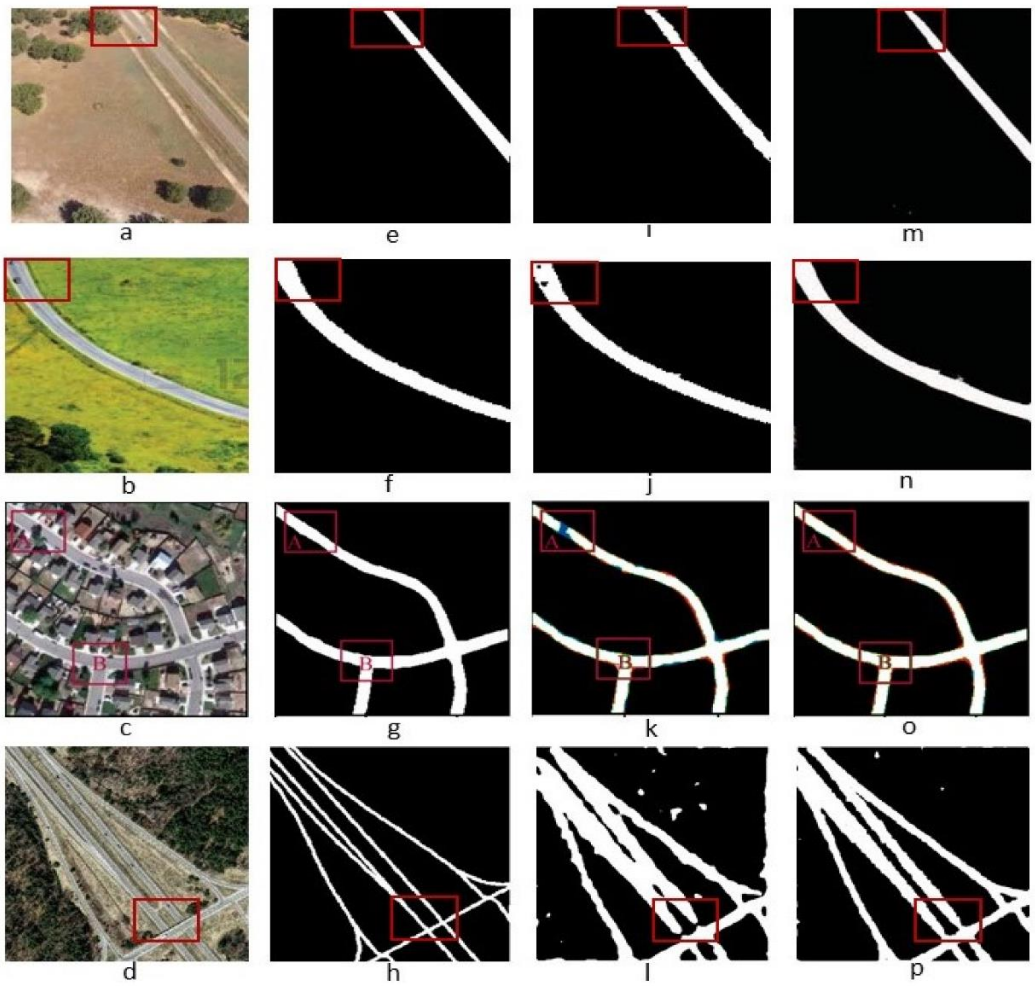
We observe that all of these tests require to be on site assessments. While these assessments are able to produce the most accurate classification of different pavement conditions, they are also very time, resource and labour intensive. Our aim is to formulate a way in which we can assess the pavement conditions remotely, which can save time, resource and labour.

Satellite imagery has emerged as a powerful tool for monitoring and understanding our planet's dynamic surface. Recent advancements in feature extraction from satellite imagery have made it a popular choice in various domains such as agriculture, environmental monitoring, urban planning and development, disaster management and much more. This project focuses on feature extraction from satellite images to classify different pavement conditions using machine learning and deep learning feature extraction algorithms.

LITERATURE REVIEW

Feature extraction from images have gained much attention in recent times due to their ability to reduce the consumptions of resources and capital. But, it does not mean that the process of extracting features as pure land features is easy. No matter what the spatial resolution of a hyperspectral image may be, the spectral signatures collected in natural environments are invariably a mixture of the signatures of various materials.

Each satellite image pixel is, according to scale, several metres to kilometers in area. Hence, each pixel is a mixture of several pure spectral signatures belonging to different geographical features such as vegetation, roads, buildings, etc. These pure spectral signatures are referred to as ***endmember***. We are only concerned with the endmembers related to pavements. We need a supervised learning algorithm which could actively separate and identify pavement-exclusive spectra from the spectral mixture from each pixel.



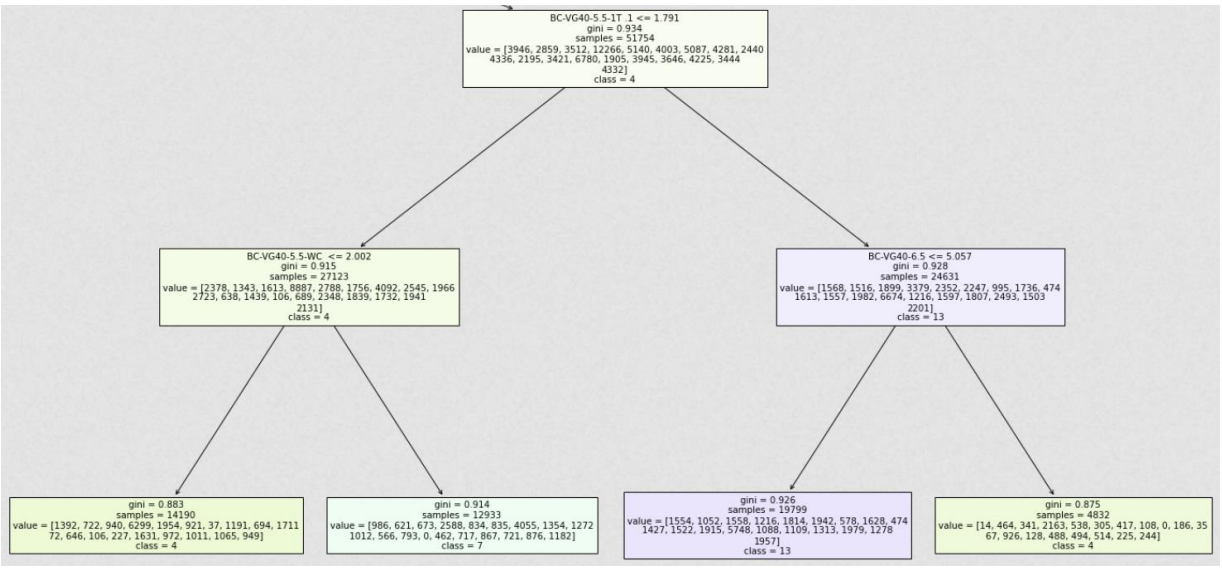
An example of road feature extraction using deep learning techniques

Decision Trees and Random Forests

Decision Trees is a popular supervised learning algorithm because they are not affected by scaling and are usually robust to irrelevant dataset features and are easy to interpret.

Decision Trees consist of a root node, intermediate nodes known as decision nodes and terminal nodes. It builds a flowchart-like tree structure where each intermediate node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node. The parameters that govern the split in a tree are discussed later in the report.



A part of one the decision trees of a random forest classifier

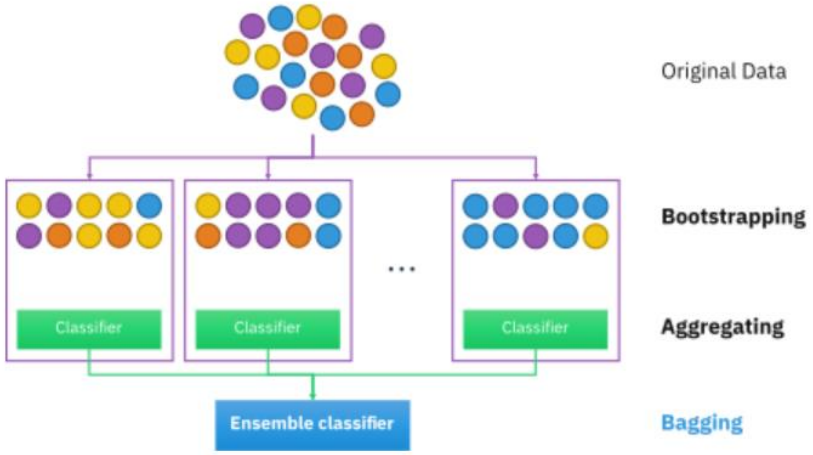
A major flaw of decision trees are that deep trees are sensitive to highly irregular patterns in the dataset which leads to ***overfitting*** on the dataset, i.e. they have low bias but very high variance. Overfitting generally means that the model has learned too deep into the patterns of the training dataset hence they perform poorly on the test dataset.

Random Forest is a way to tackle that very problem by including multiple decision trees trained on multiple splits of the same training dataset, and averaging (majority voting) those multiple deep trees. This ensures that even if some deep trees cause overfitting, they are ruled out during the final averaging or majority voting if the multiple decision trees. This may increase the bias slightly but greatly reduces the variance of the model, leading to a very high-performance model.

Random forests work on the concept of ***Bagging***, also known as bootstrap aggregation. For a training dataset, bagging repeatedly (n times) selects a dataset sample with replacement and fits trees to these samples. Steps involve

1. **Selection of Subset**: Bagging starts by choosing a random sample, or subset, from the entire dataset.
2. **Bootstrap Sampling**: Each model is created from these samples, called Bootstrap Samples, which are taken from the original data with replacement. This process is known as row sampling.

3. **Bootstrapping**: The step of row sampling with replacement is referred to as bootstrapping.
4. **Independent Model Training**: Each model is trained independently on its corresponding Bootstrap Sample. This training process generates results for each model.
5. **Majority Voting**: The final output is determined by combining the results of all models through majority voting. The most commonly predicted outcome among the models is selected.
6. **Aggregation**: This step, which involves combining all the results and generating the final output based on majority voting, is known as aggregation.

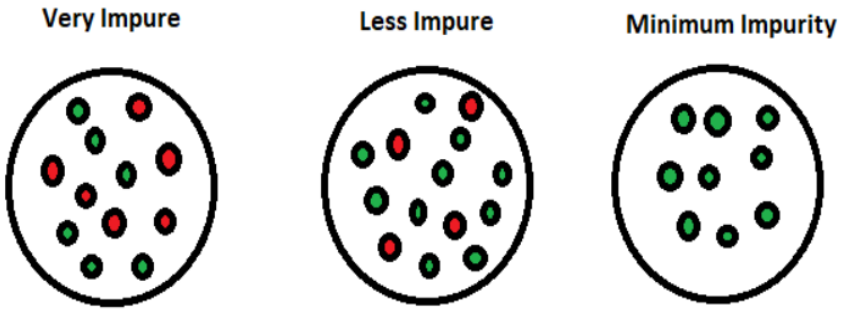


A simple representation of a random forest classifier

Random forests also include another type of bagging scheme: they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of features. This process is sometimes called "feature bagging." This is to avoid the correlation of features and trees for a bootstrap sample. If one or a few features are very strong predictors for the target variable, these features will be selected in many of the trees, causing them to become correlated. This also helps prevent the curse of dimensionality, which arises due to the high number of training features in the dataset.

Parameters of a random forest classification model:

- ***max_features***: maximum features a random forest model is allowed to train in an individual tree. Increasing max_features decreases the speed of the model but improves the performance of the model.
- ***n_estimators***: the number of trees you want to build before taking the maximum voting or averages of predictions. A higher number of trees give you better performance but makes your code slower.
- ***min_samples_leaf***: minimum number of samples required to be a leaf node. A lower number of samples will be more sensitive to small changes in the data.
- ***min_samples_split***: minimum samples required to split a node
- ***criterion***: the criteria used to make a split in the node. There are two types of metrics used:
    - **Entropy**: an information theory metric that describes the impurity/uncertainty in a group of observations. It determines how a decision tree splits its data. The image below provides an example of the purity of a set of data.



In a dataset with N target variables, the entropy is calculated as

$$E = -\sum_{i=0}^{N} p_i log_2 p_i$$

, where $\boldsymbol{p_i}$ is the probability of randomly selecting a sample of class i.
Suppose a node in the tree has only one class(pure). Its entropy is 0 hence not useful for training.
Information gain is represented as

$$Information\ Gain = E_{parent} - E_{child}$$

If we split the parent into multiple child nodes, the child with the highest information is chosen as the next split.

- **Gini Index**: Gini Index, also known as Gini impurity, calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly. Gini impurity is calculated based on the distribution of class labels in that node.

$$Gini\ index = 1 - \sum_{i=0}^{N} p_i^2$$

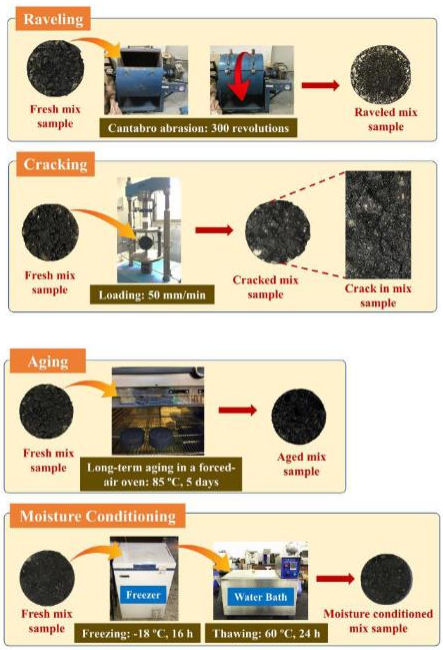
, where $\boldsymbol{p_i}$ is the proportion of samples belonging to class i in the node.
The split with the greater decrease in impurity is chosen as the better split.

DATA ACQUISITION

To acquire pure spectral endmembers for training our model, we need the pavement samples itself. The samples required were prepared at the laboratory. Different asphalt mixture specimens were created with varying proportions of bitumen and aggregate.

Before collecting hyperspectral data, some of the asphalt mixtures were treated to simulate some commonly observed distresses:

- **Raveling**: common distress observed on asphalt surfaces and involves the disintegration or loss of aggregate particles from an aggregate-asphalt matrix *(Huber, 2000;Kandhal 2016).* Under raveling distress, the fine aggregate generally wears away first, followed by the disintegration of larger fragments from the matrix.
- **Cracking**: common distress observed on flexible pavements. To simulate cracking distress in the laboratory-prepared mix specimens, a single crack was created by subjecting the sample to an indirect tensile strength testing machine and loading the specimen at 50mm/min until a single crack was formed.
- **Aging**: Due to the high temperatures involved, the film of asphalt binder present over the aggregates undergoes aging/oxidation during the production and construction of bituminous mixes, referred to short-term aging. The binder undergoes long-term aging during the pavement service *life (Kumar et al. 2022;Wang et al. 2022).*
- **Moisture Conditioning**: Asphalt-wearing surfaces are continuously exposed to traffic and environmental factors, including exposure to rain. Four moisture conditioning schemes were used to simulate similar conditions at the laboratory scale: surface wet, freeze, thaw, and a combination of the two (freeze and thaw).



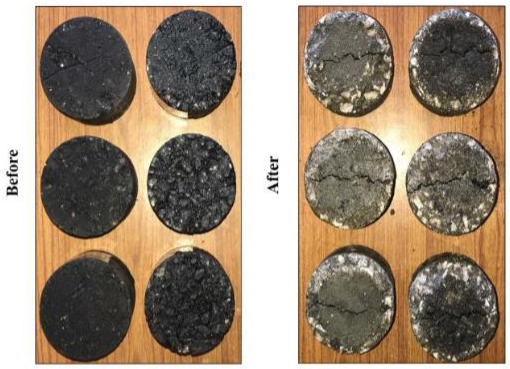


Fig. 4. Specimens before and after the Cantabro abrasion test.

Various distress simulations and specimens after distress simulation

The synthetic hyperspectral image cube is constructed from data that has been acquired from the lab which involves the spectral signatures of different laboratory prepared asphalt mixture specimens acquired with an *SVC HR-1024i spectroradiometer (Spectra Vista Corporation, Poughkeepsie, New York)*. The HR-1024i spectroradiometer covers the spectral range from around 350 to 2,500 nm using three detectors spanning across the VNIR to the shortwave infrared (SWIR1 and SWIR2) region of the

electromagnetic spectrum, with spectral sampling intervals of 1.5, 3.8, and 2.5 nm for the Si detector (VNIR), the InGaAs detector (SWIR1), and the extended InGaAs detector (SWIR2), respectively.

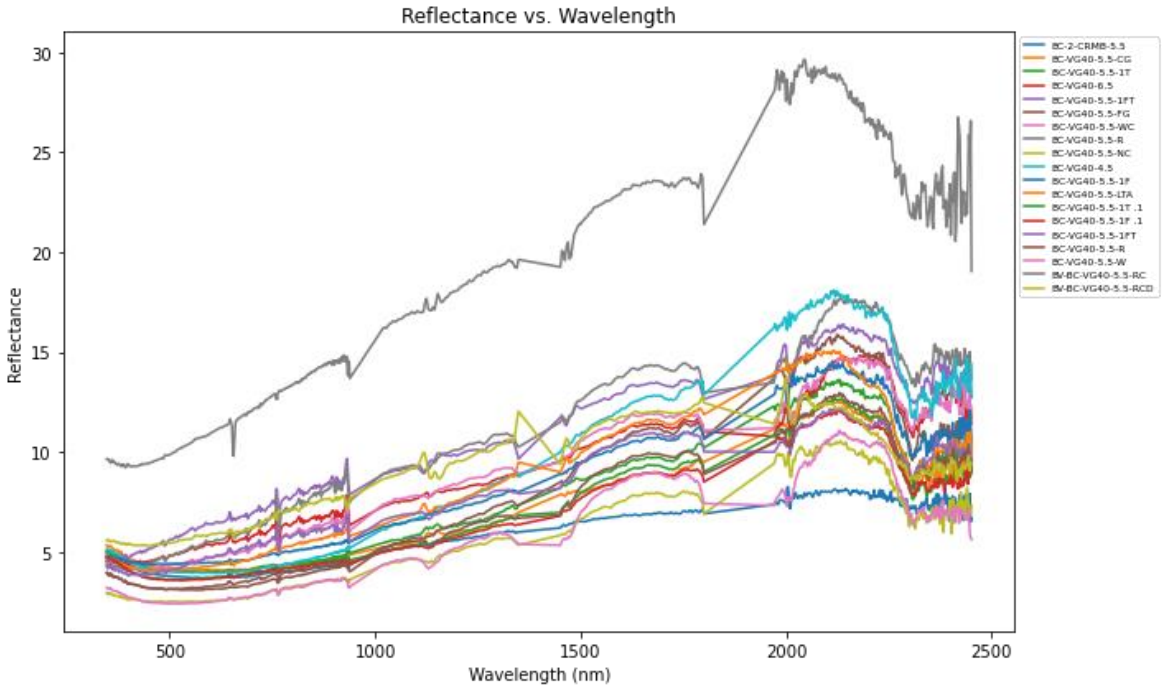
Spectra of the laboratory-prepared mix samples were acquired from a height of 1 m, with a field of view of 4° (0.003846 m2 area at the height of 1 m). Targets were measured with a set of two or three replicate spectra for each specimen, and after inspecting every spectrum for quality, suspected spectra with significant noise were rejected. Data was stored in the form of reflectance against wavelength.

The columns of the data include the wavelengths at which the spectral measurements were taken, then 19 samples whose reflectance values were given for each wavelength value. A snippet of the dataset is given below.

	Wavelength (nm)	BC-2-CRMB-5.5	BC-VG40-5.5-CG	BC-VG40-5.5-1T
0	346.8	4.96	5.33	5.04
1	348.4	4.89	5.24	4.98
2	349.9	4.87	5.28	5.03
3	351.5	4.84	5.28	5.01
4	353.1	4.86	5.26	5.00

```
data=pd.read_csv('Copy of EndMembers(1).csv', sep=',')

print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825 entries, 0 to 824
Data columns (total 20 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Wavelength (nm)     825 non-null    float64
 1   BC-2-CRMB-5.5       825 non-null    float64
 2   BC-VG40-5.5-CG      825 non-null    float64
 3   BC-VG40-5.5-1T      825 non-null    float64
 4   BC-VG40-6.5         825 non-null    float64
 5   BC-VG40-5.5-1FT     825 non-null    float64
 6   BC-VG40-5.5-FG      825 non-null    float64
 7   BC-VG40-5.5-WC      825 non-null    float64
 8   BC-VG40-5.5-R       825 non-null    float64
 9   BC-VG40-5.5-NC      825 non-null    float64
 10  BC-VG40-4.5         825 non-null    float64
 11  BC-VG40-5.5-1F      825 non-null    float64
 12  BC-VG40-5.5-LTA     825 non-null    float64
 13  BC-VG40-5.5-1T .1   825 non-null    float64
 14  BC-VG40-5.5-1F .1   825 non-null    float64
 15  BC-VG40-5.5-1FT     825 non-null    float64
 16  BC-VG40-5.5-R       825 non-null    float64
 17  BC-VG40-5.5-W       825 non-null    float64
 18  BV-BC-VG40-5.5-RC   825 non-null    float64
 19  BV-BC-VG40-5.5-RCD  825 non-null    float64
```



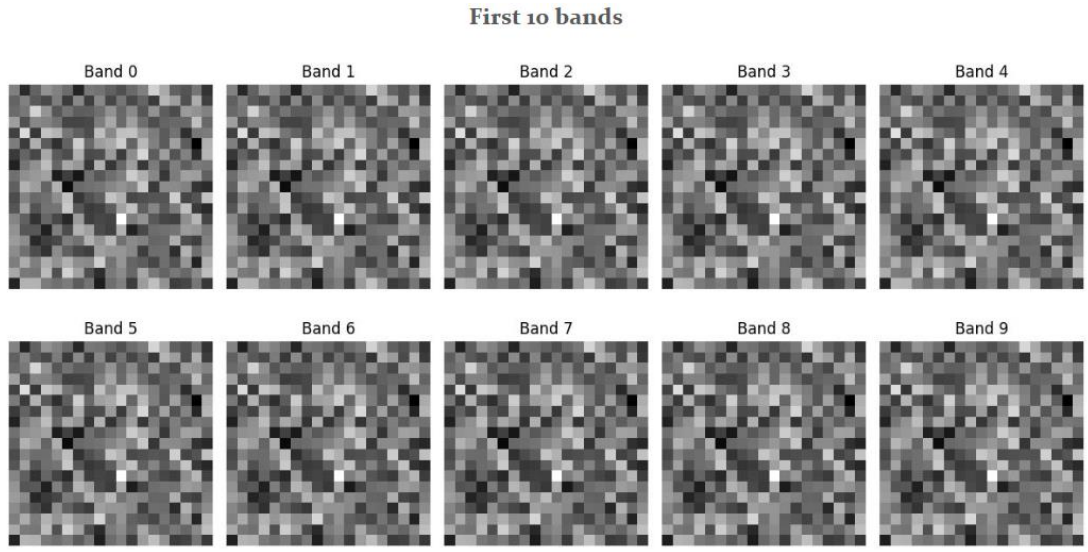
METHODOLOGY

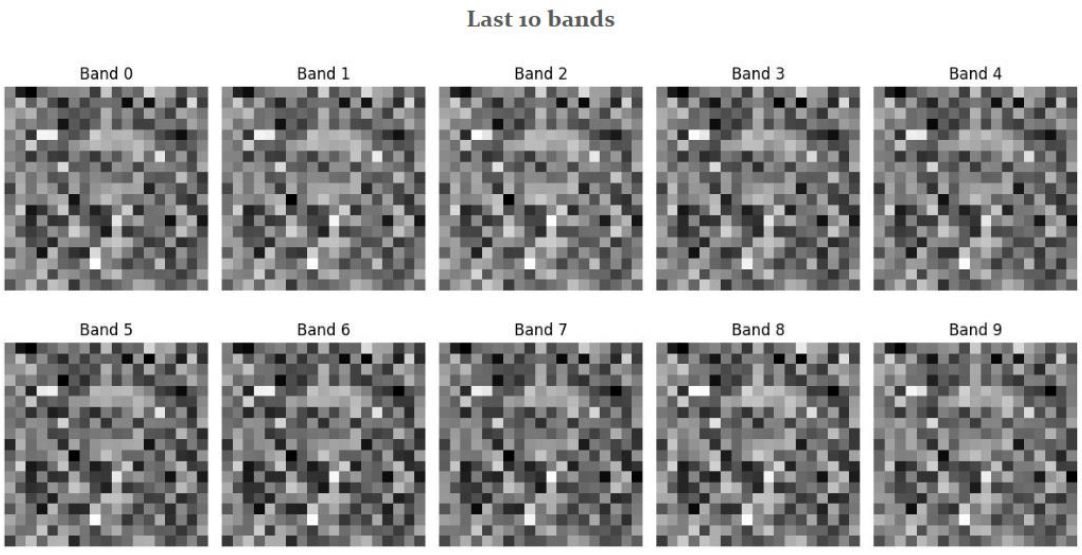
Creation of a hypercube

The process involves creation of a 19X19 size pixel image with 825 spectral bands. As the data provide has 19 samples and their spectral reflectance for 825 wavelengths. We create one pixel of one spectral band by taking the weighted average of the 19 sample reflectance for that specific spectral band. The weights have been randomly generated using numpy and between 0 and 1. We also make sure that the same weights are distributed for each pixel along all wavelengths. These weight values for all pixels are stored prop_list .

The Python code below represents the creation of the 19X19X825 size hypercube using ***numpy***:

```
matrix = np.zeros((19, 19, 825))
prop_list=[]; # list of weights for each pixel

for i in range(19):
  for j in range(19):
    mat=0.5*np.random.rand(1, 19) # making a random weight array
    prop_list.append(mat)
    sum=np.sum(mat)
    for index in data.index:

      row=data.loc[index].to_numpy()
      matrix[i][j][index]=np.dot(mat, row)/sum # weighted average of each pixel
```





Creation of training dataset

First we create a map which maps each index to their respective class

```
my_dict={0:'BC-2-CRMB-5.5', 1:'BC-VG40-5.5-CG', 2:'BC-VG40-5.5-1T', 3:'BC-VG40-6.5',
         4:'BC-VG40-5.5-1FT', 5:'BC-VG40-5.5-FG', 6:'BC-VG40-5.5-WC', 7:'BC-VG40-5.5-R',
         8:'BC-VG40-5.5-NC', 9:'BC-VG40-4.5', 10:'BC-VG40-5.5-1F', 11:'BC-VG40-5.5-LTA',
         12:'BC-VG40-5.5-1T .1', 13:'BC-VG40-5.5-1F .1', 14:'BC-VG40-5.5-1FT', 15:'BC-VG40-5.5-R',
         16:'BC-VG40-5.5-W', 17:'BV-BC-VG40-5.5-RC', 18:'BV-BC-VG40-5.5-RCD'}
```

Now to create the training dataset, we take one proportion list from prop_list take the maximum proportion from the list. The index where the max proportion lies is the index of the respective class.

An example is shown below where one proportion list is taken, the index with max proportion is taken and the class mapped to that index is taken.

```
index=np.argmax(prop_list[0])
print("maximum index: ", index)
print("distress condition related to that index: ", my_dict[index])
```

```
18
BV-BC-VG40-5.5-RCD
```

Now we do the following steps:

1. Create a empty dataframe ***new_data***
2. Traverse every list in the prop_list
3. Find the max proportion and the corresponding distress condition of that list
4. Create a copy of endmember dataframe
5. Traverse every row in that dataframe
6. Multiply each member in the list to each member in a row(in a weighted sum fashion)
7. Now create a new column in the dataframe and assign that full column with the corresponding distress condition
8. Append that modified dataframe to ***new_data***

```
new_data=pd.DataFrame()                                          # create new empty training dataframe

for prop in prop_list:                                           # do it for every proportion list
  max_index=np.argmax(prop)                                      #find max proportion index
  distress=my_dict[max_index]
  dat=copymaster.copy()                                          # create a copy of the endmember dataframe
  for index in dat.index:                                        # traverse each row
    dat.iloc[index, 1:]=dat.iloc[index, 1:]*prop.reshape(-1)     # multiply each proportion to each reflectance in one row
  new_col=np.full(len(dat), distress)
  cat=np.full(len(dat), max_index)
  dat['Distress']=new_col                        # assign new column with that max proportion distress in the endmember dataframe
  dat['Category']=cat
  new_data=pd.concat([new_data, dat], ignore_index=True)     # append the modified endmember datadrame to the training dataframe
```

Since we traverse every list in prop_list of size(19X19) 361 and append the same amount of endmember dataframes of size 825, the size of the ***new_data*** dataframe is

$$\mathbf{825 \times 361 = 297825\ rows}$$

In the dataframe, we have two new columns Distress and Category, where Category is the encoding of the Distress column which is the corresponding index mapped to that Distress condition. This Category column will be the target variable for our training dataset since Distress column is not a numerical column.

Training the model

To train the model we first split our training dataset, into a training set and a test set. We also define our training and target variables.

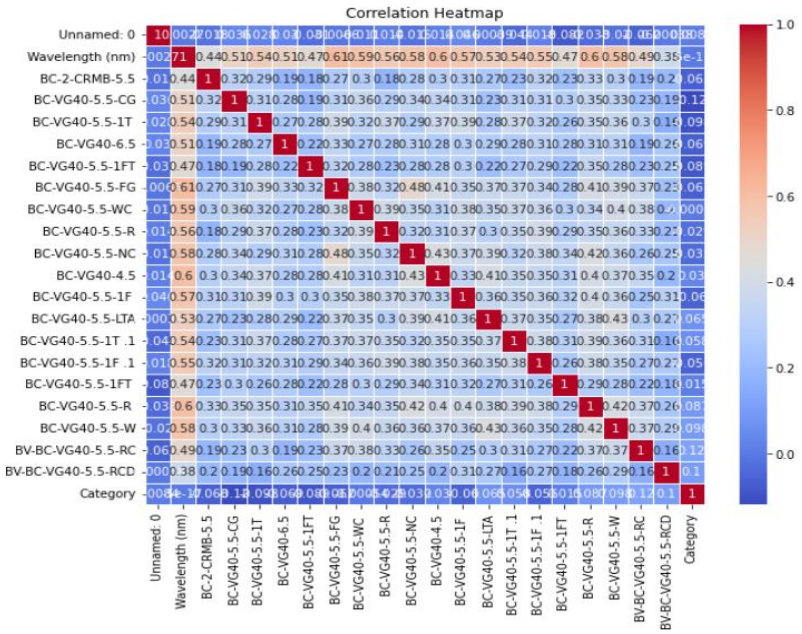
```
from sklearn.model_selection import train_test_split

y=train_data['Category'].values
X=train_data.drop(columns=['Category', 'Distress', 'Unnamed: 0'])
y2=test_data['Category'].values
X2=test_data.drop(columns=['Category', 'Distress', 'Unnamed: 0'])

X_train1, X_test1, y_train1, y_test1=train_test_split(X, y, test_size=0.3, random_state=42)
X_train2, X_test2, y_train2, y_test2=train_test_split(X2, y2, test_size=0.3, random_state=42)

X_train=pd.concat([X_train1, X_train2], ignore_index=True)
X_test=pd.concat([X_test1, X_test2], ignore_index=True)
y_test=np.concatenate((y_test1, y_test2))
y_train=np.concatenate((y_train1, y_train2))
```

The model that we are implementing is a random forest model. We have used the ***scikit-learn*** library in Python to implement a ***RandomForestClassifier*** model. Since it is a hassle to repeatedly train our model with different parameters, we shall implement hyperparameter tuning since Random Forest takes more time to train a model. We have implemented randomized search cross validation, which in contrast to grid search cross validation, does not evaluate all possible combinations of hyperparameters, rather it evaluates only a subset of the possible combinations in order to find the classifier with the best parameters and accuracy.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

# Create a Random Forest Classifier
rf_classifier = RandomForestClassifier()

# Define the hyperparameters and their possible values to search
param_dist = {
    'n_estimators': np.arange(100, 601, 100),
    'max_depth': list(np.arange(10, 61, 10)),
    'min_samples_split': np.arange(2, 9),
    'min_samples_leaf': np.arange(1, 4),
}

# Create the GridSearchCV object
r_search = RandomizedSearchCV(estimator=rf_classifier, param_distributions=param_dist, n_iter=10, cv=5, scoring='accuracy',
                           verbose=1, n_jobs=-1)

# Fit the RandomizedSearchCV to the data
r_search.fit(X, y)

# Print the best hyperparameters found
print("Best Hyperparameters:", r_search.best_params_)

# Get the best estimator (model with the best hyperparameters)
best_rf_classifier = r_search.best_estimator_

# Evaluate the model on the test set
accuracy = best_rf_classifier.score(X_test, y_test)
print("Test Set Accuracy:", accuracy)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best Hyperparameters: {'n_estimators': 200, 'min_samples_split': 6, 'min_samples_leaf': 3, 'max_depth': 20}
Test Set Accuracy: 0.6086354634956687
```

We observe that our model does not predict well with an accuracy of 60% on the test dataset, which can be hinting that there might exist some correlation between features. The correlation heatmap suggests that the wavelength has a very high correlation with the other features in the dataset.



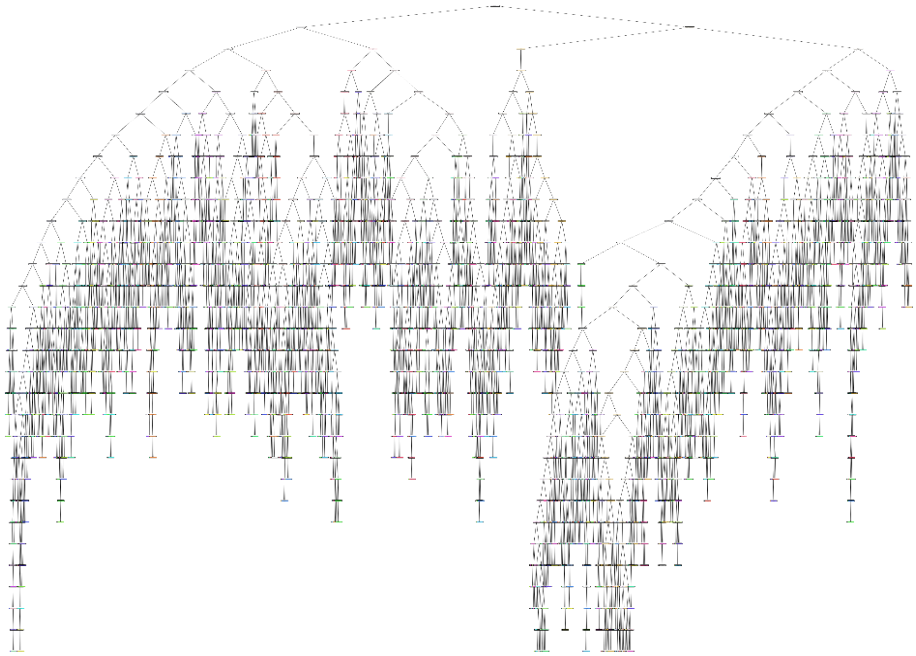
Now we remove the wavelength feature from the training set and fit the model again with hyperparameter tuning.

```
rf_classifier = RandomForestClassifier()

# Define the hyperparameters and their possible values to search
param_dist = {
    'n_estimators': np.arange(100,601, 100),
    'max_depth': list(np.arange(10, 61, 10)),
    'min_samples_split': np.arange(2, 8),
    'min_samples_leaf': np.arange(1, 4),
}

# Create the GridSearchCV object
rf_search = RandomizedSearchCV(estimator=rf_classifier, param_distributions=param_dist, n_iter=10, cv=5, scoring='accuracy',
                               verbose=1, n_jobs=-1)

# Fit the RandomizedSearchCV to the data
rf_search.fit(X_train, y_train)

# Print the best hyperparameters found
print("Best Hyperparameters:", rf_search.best_params_)

# Get the best estimator (model with the best hyperparameters)
best_rf_classifier = rf_search.best_estimator_

# Evaluate the model on the test set
accuracy = best_rf_classifier.score(X_test, y_test)
print("Test Set Accuracy:", accuracy)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best Hyperparameters: {'n_estimators': 100, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_depth': 60}
Test Set Accuracy: 1.0
```

Now we can observe that the model fits well on the dataset. We see that the model predicts the leading feature condition with 100% accuracy on the test data. We observe that removing the wavelength feature has helped fit the model better.

Although plotting tree of one of the estimators of the classifier is not intuitive and very complex to understand due to its number of features and depth, one example of one of the decision tree has been plotted below.



RESULTS

- We were able to successfully create a random synthetic hyperspectral cube where each pixel was a mixture of random proportions of various samples.
- We created a random forest classification model and trained it on the hypercube dataset. We also implemented hyperparameter tuning on the classifier using ***RandomizedSearchCV*** to achieve the best performing classifier parameters.
- The classifier was able to predict with 100% accuracy on the test after removing the highly correlated wavelength feature. This means, that the classifier is **not** overfitting on the training dataset.

The detailed code for the project is available on the GitHub link here.

WORK TO BE DONE

- We have successfully created the model for classifying the leading distress condition in a mixture of random proportions of pavement conditions. Our main goal is to extract the pure endmembers from a mixture of different spectral endmembers other than those of asphalt mixtures such as vegetation, water, etc.
- The next phase of the project would involve use of random forest model to try and classifiy on a real hyperspectral image, and then we gradually delve into much more sophisticated methods which involve use of deep learning techniques and various unmixing algorithms.

REFERENCES

- Patel, V. D., Kumar, A., Bharti, R., Choudhary, R., & Kumar, A. (2023). Evaluation of Spectral Characteristics of Asphalt Mixtures. *Journal of Materials in Civil Engineering*, *35*(9), 04023295
- Wang, J., & Chang, C. I. (2006). Applications of independent component analysis in endmember extraction and abundance quantification for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, *44*(9), 2601-2616.
- Martin, G., & Plaza, A. (2012). Spatial-spectral preprocessing prior to endmember identification and unmixing of remotely sensed hyperspectral data. *IEEE journal of selected topics in applied earth observations and remote sensing*, *5*(2), 380-395.
- *https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/*
- *https://scikit-learn.org/stable/modules/tree.html*
- *https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*
- *https://www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/*