# Classification of Soil Using Multi-Spectral Imagery From the Landsat Satellite

Alexander L. Leven[1] and Dante A. Sanaei[2]
*Purdue University School of Aeronautics and Astronautics, West Lafayette, Indiana, 47906, USA*

Prof. Veeraraghava Raju Hasti[3]
*Purdue University School of Mechanical Engineering, West Lafayette, Indiana, 47906, USA*

***Abstract*—The following report is an analysis of the application of artificial intelligence to the field of satellite remote sensing. Given a large dataset of pixel values and their corresponding soil classification, the report evaluates the accuracy and effectiveness of several different classification models. Through the analysis of vital performance metrics, two models are chosen and are then utilized in finding key trends in the dataset. Overall, the mission of this paper is to determine which models are most desirable and what discoveries are hidden within the Landsat image data.**

***Keywords*—*Classification, K-Nearest Neighbors, Remote Sensing, Support Vector Machines***

## I. INTRODUCTION

In the design of observation satellites, one of the key fields of interest is in remote sensing. To summarize, remote sensing essentially allows for the acquisition of information on the surface of the Earth without making any in-situ physical measurements. Using reflected solar radiation propagated from a location of interest, a satellite's detector is able to detect and analyze such back-scattered data. Such applications of remote sensing include, among numerous other examples, the ability for satellites to monitor meteorological changes, the atmospheric environment, or topographic conditions [1].

In this project, the team is using Landsat data obtained by the Australian Centre for Remote Sensing. The Landsat satellite is capable of obtaining a significant amount of information based on a specific imaged scene. Here, the satellite data is taken from four digital images of the same area using different spectral bands. Specifically, the data of interest that the paper is examining is a series of 9 pixels across these spectral bands generating 36 distinct values. With these individual pixels, a label has been given to each set corresponding to a particular soil characteristic: red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, and very damp grey soil. Thus, this soil type is defined as the one independent variable in the dataset while the 36 pixel values are the dependent variables [2].

[1] Student, School of Aeronautics and Astronautics
[2] Student, School of Aeronautics and Astronautics
[3] Project Advisor, School of Mechanical Engineering

The overall objective of the project is to analyze the dataset and classify a given pixel with the likely type of soil that is represented. Beyond the overall results of the classification algorithm, the team will also be looking to evaluate different artificial intelligence models that could classify the dataset most accurately and efficiently. Namely, the six methods involved are logistic regression, support vector machines (linear, polynomial, and RBF), Naive Bayes, and K-Nearest Neighbor (KNN). For each of these methods, a model will be developed, and the classification results of the test data will be analyzed. Specific performance metrics such as execution time, precision, recall, F1-score, support and overall accuracy will be identified and studied. Once all models have been developed, a specialized scoring method will be utilized to determine the two most effective classification models. These will be utilized for the next portion of the report.

An additional objective of the analysis is to determine any patterns within the dataset after using the AI method (i.e. if the AI reveals anything that could not be seen with inspection). In terms of understanding the AI system, it may be useful to note what classification devices may arise from running multiple tests. The two models that are identified in the initial model analysis will be used to find hidden trends and other important findings that can not be observed through the dataset alone.

Overall, the introduction of Artificial Intelligence to the analysis of this multi-spectral data is vital because it can be incredibly difficult to develop a purely mathematical algorithm with no training data that is accurate enough to classify soil imagery; this is due to the fact that the images that are being taken are consistently unique. Soil is not always perfectly unobstructed and clear for satellite imagery, and imperfections are always evident. Therefore, it is crucial to develop an artificial intelligence classification model that utilizes massive amounts of training data in order to identify difficult and unclear soil imagery.

If the project is generalized to a classification problem, then one can claim that the project does not add any novel algorithmic approaches to such problems. However, the importance of what this problem does is provide a means to interpreting remote sensing data. Currently, there are numerous private companies such as Space Know or Astro Digital which rely on data and imagery taken from remote sensing satellites to analyze optimal agricultural yields. Such company objectives are similar to the impact of this project. That is, using a soil classification algorithm could help determine which subplots of a given imaged land are arable or usable land for farming [3]. In an ever-growing world where fertile land becomes an increasingly useful commodity, such techniques to improve agricultural efficiency are necessary to ensure reliant crop growth for a given population.

## II. Literature Review

[4] **Remote sensing for agricultural applications: A meta-review**

This paper discusses how remote sensing has the ability to significantly improve the current state of agricultural practices. Specifically, by analyzing soil and plant types, the paper discusses how remote sensing can be used to ascertain important qualities for planting crops for certain seasons. The relevance of this paper to the project at hand is one of contextualization. Since our project is dealing with the use of AI methods to classify soil samples in the means to demonstrate AI's necessity in remote sensing, understanding a more specific view at how remote sensing is actually used is helpful foundational knowledge. The database of the research covers a wide range of agronomic factors that can be measured by satellite including green cover, vegetation temperature, and soil moisture among other things. By employing a large array of empirical methods to analyze such a dataset, the paper demonstrated how the analysis of such data can lead to wide-ranging applications in land-use forecasting and precision farming. The effects of the research show the reliability and use for remote sensing systems to make a significant impact on the future of the agriculture community.

[5] **Deep learning in remote sensing applications: A meta-analysis and review**

With the rise of remote sensing satellite technologies, deep learning algorithms have seen a recent rise of increased usage. This paper seeks to conduct an analysis on the development of such AI algorithms and their applications to the remote sensing field. In particular, this paper helps to provide a contextualized framework for understanding the importance of an AI learning algorithm for remote sensing purposes. As mentioned earlier in the introduction, numerous companies have employed AI solutions for remote sensing data in the agricultural community. As such, this paper allows the team to get a more detailed view at the field on a wide-scale -- that is, encompassing many factors in how AI affects the industry. The overall results of the paper detailed what AI algorithms were most commonly used in such remote sensing problems like the farming industry. The paper also made the claim that recent deep learning techniques have shown more precision compared to traditional techniques such as support vector machines (SVM). Since the team plans to test multiple AI methods learned throughout the course, this claim is asserted by Ma et. al. is one that can now be looked into with more scrutiny. Further detail in the paper has given the team more course of action to look for certain results when the AI model is created in the future.

[6] **From Smart Farming towards Agriculture 5.0: A Review on Crop Data Management**

This paper is a continuation of the research that was done in investigating the use of AI and remote sensing in agricultural applications. Specifically, in this paper, the authors focus their study on comparing traditional data acquisition techniques to current methods utilizing remote sensing/deep learning to help growers make optimized and sustainable farming decisions. In addition to the previous documents, this paper gives more contextual background to the issue at hand regarding the use of AI in crop management. The article posits that the growth of AI and other relevant technologies could be one of the most advantageous developments in the past 30 years for the agricultural industry. Hence, this paper furthered our understanding of the topic in that it shows that the research we are conducting is of a significant priority. Though the project's own AI remote learning may be on a smaller scale, it is entirely representative of a major solution that could aid one of the US's largest industries -- our project is merely an entry point into such applicable topics.

## III. Description of Dataset

The database consists of multi-spectral values of pixels obtained from a small area of an imaged scene. Such pixels are further derived from a 3x3 neighborhood within this satellite image. Also, recall that each pixel within the database corresponds to a value within 0 to 255. Moreover, for each row of pixels, there is a classification label associated with the central pixel of each 3x3 neighborhood. Then for each neighborhood of 9 pixels, there are four spectral bands to yield a total of 37 columns in the dataset (36 pixels and 1 classification label). As mentioned, the 36 pixels comprise the dependent variables and the classification label is the independent variable of the experiment.

To restate the problem description, the goal now is that given these 36 pixels to predict the classification number. To start, the given dataset has already been split into training and testing sets: training composed of 4435 instances and testing composed of 2000 instances [2]. Within the dataset, there are six different types of soil classes -- the labels of which are shown below:

**Table 3.1. Classification Labels (From Ref. [2])**

| Label | Class |
|---|---|
| 1 | Red Soil |
| 2 | Cotton Crop |
| 3 | Grey Soil |
| 4 | Damp Grey Soil |
| 5 | Soil w/ Vegetation Stubble |
| 7 | Very Damp Grey Soil |

## IV. Methodology

For this project, a key objective is to successfully determine soil type from the pixel values for a small image area; therefore, a classification model must be developed to automate the analyzation process. Classification is a broad subject within machine learning, and several different types of classification models exist. In the project, four methods were used: logistic regression, support vector machines (SVM), Naive Bayes, and K-Nearest Neighbor. It is vital to analyze the results of these different strategies, and determine which produce the most favorable outcome. The following is a discussion and overview for each method. Moreover, for each case, the group intends to produce a confusion matrix heatmap and classification reports to display the results of the method.

After analysis is performed on each classification model, two will be chosen based on predetermined criteria, and additional analysis will be performed on the finding within the dataset. In this section of the report, key trends that have been discovered from the models will be evaluated through visuals and data.

## A. Classification Model Development

### *1. Logistic Regression*

Logistic regression is a type of classification that predicts the categorical dependent variable using a set of independent variables. By using a logistic function, the model will classify a dependent variable represented by an indicator variable (such as 0 or 1). In the case of this project, there are six possible classifications of soil which requires six dependent variables; thus, multinomial logistic regression is required [10][7].

The hyperparameters required in our logistic regression model are as follows: penalty, inverse of regularization strength, fit intercept, and max iterations. These have been tuned in an iterative method in order to maximize the accuracy score and reduce execution time, and the values can be seen in the table below.

**Table 4.1. Logistic Regression Hyperparameters**

| Method | Penalty | Inverse of Regularization Strength (c) | Fit Intercept | Max Iterations |
|---|---|---|---|---|
| **Logistic Regression** | “12” | 1.0 | True | 3000 |

The Hyperparameters in the table above are relatively normal for Logistic Regression, but the maximum iterations has been increased to 3000 in order to enhance the accuracy of the model.

The code for this model is relatively simple. Essentially, the Landsat training data is used to create and fit the regression model, then we use this model to predict dependent variables in the test dataset. The classification report and the confusion matrix heatmap are then printed for a better understanding of the model’s performance.

The main issue with the logistic regression model is that it can possibly force the execution time of the program to grow to unacceptable lengths. In order to limit this, there is a hyperparameter of maximum iterations that will forcibly stop the model at that point. By having a lower maximum iteration value, the execution time decreases, but the accuracy of the model also decreases. To mitigate this issue, experimentation of scaling of the data can be performed. Additionally, a significant disadvantage to logistic regression is on high dimensional datasets, the model may overfit the training set, and it can also be difficult to capture some of the complex relationships that are present in the 36 independent variables in the dataset.

### *2. Support Vector Machine*

Support vector machines (SVM) is another very popular method for the classification of independent variables, and it is a great option to experiment with for our datasets. The SVM is a generalization of the maximal margin classifier which has proven to be very simple and robust. Just like logistic regression, the SVM model classifies a dependent variable represented by an indicator variable (such as 0 or 1). In the case of this project, there are six possible classifications of soil which require six dependent variables; thus, higher dimensional SVM is required [8]. SVM is unique in the fact that there are several possible kernels that can be utilized and produce differing performance parameters. Linear, Polynomial and Radial Basis Function were chosen as three kernels that would be separately developed and evaluated [10].

The hyperparameters required in our SVM model are as follows: kernel, degree, and maximum iterations. As mentioned previously, there were three kernels types that were separately attempted (linear, polynomial, and radial basis function). The rest of the parameters (degree and max iterations) have been tuned in an iterative method in order to maximize the accuracy score and reduce execution time. The hyperparameters can be seen in the table below.

**Table 4.2. Support Vector Machine Hyperparameters**

| **Method** | **Kernel** | **Inverse of Regularization Strength (c)** | **Degree** | **Max Iterations** |
|---|---|---|---|---|
| **SVM (Linear)** | “linear” | 1.0 | 64 | 500 |
| **SVM (Polynomial)** | ‘poly’ | 1.0 | 2 | 1000 |
| **SVM (RBF)** | ‘rbf’ | 1.0 | 2 | 1000 |

As we see in the above table, the maximum iterations for the linear kernel were decreased by fifty percent when compared to the polynomial and rbf kernels. This was done in order to lower the excess execution time. Warnings for hitting the maximum number of iterations were given for both the linear and polynomial kernel models, but these warnings were ignored since the accuracy was in a satisfactory range.

This classification algorithm works by utilizing the training data to create and fit the SVM model, then we use this model to predict dependent variables in the test dataset. The classification report and the confusion matrix heatmap are then printed for a better understanding of the model’s performance.

Just as in logistic regression, the main issue that SVM faces is that it can possibly force the execution time of the program to grow to unacceptable lengths. In order to limit this, there is a hyperparameter of maximum iterations that will forcibly stop the model at that point. By having a lower maximum iteration value, the execution time decreases, but the accuracy of the model also decreases. To mitigate this issue, experimentation of scaling of the data can be performed. Additionally, choosing the best kernel function is not straightforward, and it can only be done by manually attempting all three (thus increasing execution time).

### *3. Naive Bayes*

The Naive Bayes classifier is the third method that is used in the project. Essentially, Naive Bayes makes the assumption that the relevant features of a given measurement are independent of one another. Since this assumption is necessarily strong and the “independent” conditions are generally not physically independent, the application of Bayes theorem is “naive.” Just like logistic regression and SVM, the N-B model classifies a dependent variable represented by an indicator variable (such as 0 or 1) [10]. In the case of this project, there are six possible classifications of soil which requires six dependent variables; thus, higher dimensional N-B is required.

The Naive Bayes model does not require any hyperparameters.

The Naive Bayes model is developed in a very similar manner to both SVM and Logistic regression, as it develops and fits a model with the training dataset. A confusion matrix and classification report are then generated by using the model to predict the soil classifications in the test dataset.

The Naive Bayes model does not have the same execution time issues that were faced in the SVM and logistic regression models, but its main issue is that the strong assumption about feature independence does not hold true for real data.

### *4. K-Nearest Neighbor*

The last method tested in the project is the K-Nearest Neighbor (KNN) algorithm which employs the use of instance-based learning. In other words, learning algorithms of this type use a training instance set and then compare the new problem instances against the given training set. For the nearest neighbor algorithm, this method is essentially model-free and is typically used for classification and pattern recognition -- both of which are desirable traits for this project [10]. Moreover, it would be expected for such methods to produce highly efficient and accurate data without providing much insight into the nature of the patterns between data. As such, the KNN method partly achieves the project's objective in producing an accurate classifier but cannot give any further information beyond that [9]. For this reason, it would be expected in the final results to rely on the outcomes from the methods previously mentioned.

The K-NN model does not require any hyperparameters.

The algorithm for this method is slightly different than the previous three models. Because it is important to choose a K value that provides the best performance parameters, we must first iterate through several possible K inputs. After a K value with the highest accuracy score is found, the model is created and fit with the training data, then predictions will be made with the test data.

The K-NN model also has a problem with execution time, as the iterative method of finding a K value requires a different model to be generated several times. This can add a huge chunk of time to the total execution of the program. In order to mitigate these issues, we can only run the iterative K value function only one time, or scale the data so the fitting and predicting takes less time.

## B. Landsat Image Data Evaluation

After thorough analysis of the results and performance metrics for the six classification models that were developed, two will be chosen as the most optimal. Now we will transition from investigating the effectiveness of different classification methods to the evaluation of the provided Landsat dataset.

Using both chosen methods, we will plot a bar plot of the predicted soil classifications. This chart will allow better visualization of the accuracy of the results and which soil types created the most issues for the models.

Issues and difficulties that are identified will be further investigated by removing all high scoring soil types. Thus, we will generate two new confusion matrices that will only consist of classifications that had significantly lower correct predictions.

## V. Results and Discussion

### A. Classification Model Analysis

As presented, a number of different classification models were run using the same dataset. The objective, then, was to determine an optimal method that performs well in terms of precision, accuracy, and response time. In the following sections, the performance results of each individual method will be explored in greater detail including notes on confusion matrices and accuracy scores. In the final section, each method will be compared against one another to assess which model performed the best. As there were multiple metrics used to determine the capability of a method, a weighted total performance score was developed to account for qualities deemed more important.

#### *1. Logistic Regression*

Beginning with the logistic regression method, the confusion matrix and metrics are displayed below,

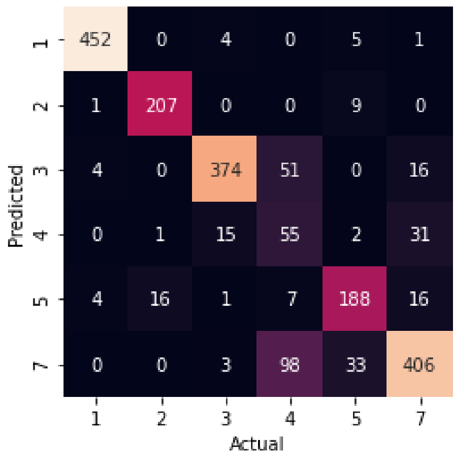


**Figure 5.1. Logistic Regression Confusion Matrix**

**Table 5.1. Logistic Regression Metrics**

| Metrics | Values |
|---|---|
| Average Execution Time[4] | 4.95 s |
| Accuracy | 84.1% |

The confusion matrix in Fig. 5.1 demonstrates that the model was accurate in its classification. For most classification labels, the method was able to accurately predict a given test set against the actual label in a relatively slow amount of time. The method's accuracy, however, is with exception to Label 4 corresponding to damp grey soil. Here, the confusion matrix shows that the method was only able to accurately classify 35% of the damp grey soil values. Looking closer, the reason for this inaccuracy becomes more apparent in that the other two labels the algorithm incorrectly

[4] Average execution time was compiled through several runs of the code block on an average laptop. While execution times will vary based on many conditions, we assume that the value obtained is indicative of overall performance of the model.

predicted were "grey soil" and "very damp grey soil." Considering that the other three classification labels are visually completely different, it is expected that the algorithm would be far more accurate in those other cases. In other words, the visual differences between, for example, red soil and cotton crop is more apparent than damp grey soil and very damp grey soil. Hence, moving forward, a key factor in determining the quality of a classification method is the level for which it can accurately distinguish between the three grey soil types.

### *2. Linear SVM*

Next, the confusion matrix and metrics for the Linear Support Vector Machine method is displayed below,

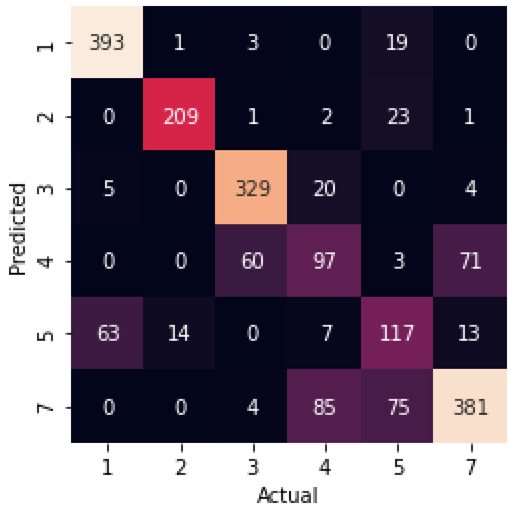


**Figure 5.2. Linear SVM Confusion Matrix**

**Table 5.2. Linear SVM Metrics**

| Metrics | Values |
|---|---|
| Average Execution Time | 219 ms |
| Accuracy | 76.3% |

The confusion matrix in Fig. 5.2 demonstrates that the model was less accurate in its classification than the logistical regression method (by about 8%). Upon further examination, however, the Linear SVM not only was not accurately classifying the grey soil labels, but also the cases of red soil and soil with vegetation stubble. For the red soil cases, about 10% of the cases were inaccurately predicted to be soil with vegetation stubble. Then, for the actual vegetation soil cases, the model inaccurately predicted nearly half of them to be very damp grey soil. So, even though the model has a relatively overall good accuracy score of 76.3%, there are problematic issues on an individual level for specific soil types. This is as opposed to the previous logistic regression matrix which appeared to only have issues with the grey soil cases.

*3. Polynomial SVM*

Next, the confusion matrix and metrics for the Polynomial Support Vector Machine method is displayed below,

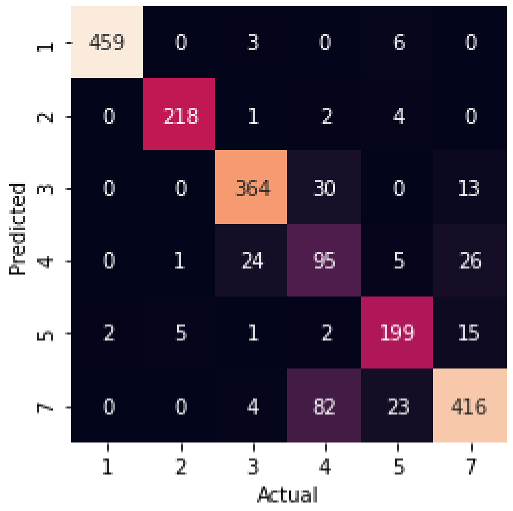


**Figure 5.3. Polynomial SVM Confusion Matrix**

**Table 5.3. Polynomial SVM Metrics**

| Metrics | Values |
|---|---|
| Average Execution Time | 394 ms |
| Accuracy | 87.5% |

The confusion matrix in Fig. 5.3 demonstrates that the model has slightly more accuracy than the logistic regression method shown in Fig. 5.1 (by about 3%). Also, compared to the previous method of the same SVM family -- the linear model -- the polynomial method has significantly better accuracy. Upon further examination, moreover, the only issues that the polynomial SVM suffered was again with the damp grey soil classification. For these cases, the model predicted about half of the actual damp grey soil to be correct while the other half was classified as very damp grey soil. In addition to being slightly more accurate than the logistic model, the execution time is much faster (350 ms compared to 4.95 s). Thus, when considering both of these metrics, the polynomial SVM is far more optimal.

*4. Radial Basis Function SVM*

Next, the confusion matrix and metrics for the Polynomial Support Vector Machine method is displayed below,

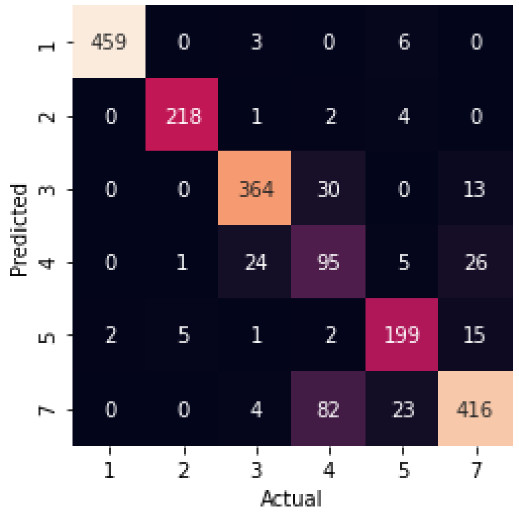


**Figure 5.4. Radial Basis Function SVM Confusion Matrix**

**Table 5.4. Radial Basis Function SVM Metrics**

| Metrics | Values |
|---|---|
| Average Execution Time | 350 ms |
| Accuracy | 87.5% |

Note that when comparing the confusion matrices of the radial basis function (RBF) to the polynomial SVM methods, they are found to be identical. Since the values comprising the dataset are as complex as other sets, the similarity between the two methods may have been expected. The one key difference that is noted, however, is that the RBF method is about 12.6% faster in terms of execution time. Thus, when selecting an optimal solution, the RBF method would be better because the delayed computation time of Polynomial SVM is a detriment.

*5. Naive Bayes*

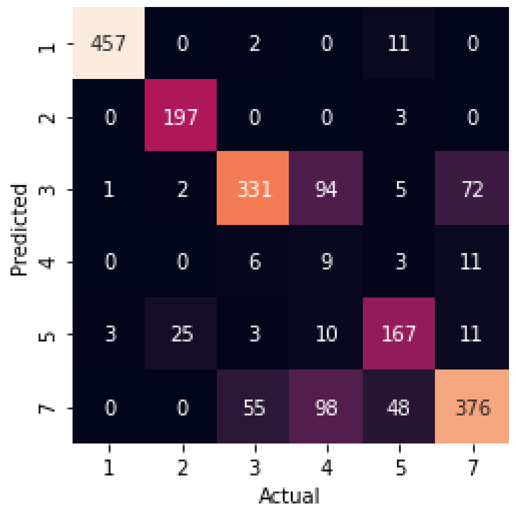


**Figure 5.5. Naive Bayes Confusion Matrix**

**Table 5.5. Naive Bayes Metrics**

| Metrics | Values |
|---|---|
| Average Execution Time | 49 ms |
| Accuracy | 76.9% |

As shown above in Fig. 5.5 and Table 5.5, though Naive Bayes performs the quickest out of the methods being tested, it is one of the most inaccurate. Compared to linear SVM (which was 0.6% less accurate), Naive Bayes performs worse in classifying the grey soil -- which as mentioned, is an area that would be the hardest for the methods to accurately classify. However, as seen in Fig. 5.5, for the label corresponding to damp grey soil, the accuracy was only 8%. In this case, Naive Bayes inaccurately predicted more of these cases to be grey or very damp grey soil. Then regarding grey and very damp grey soil, Naive Bayes's performance was again limited -- though less so than damp grey soil. For these two cases, Naive Bayes was able to classify with about 70% accuracy which compared to the higher accuracies in red soil (98%), is only moderately good. Naive Bayes is less computational expensive compared to other classifiers, so the quick response time is a result; however, the accuracy as measured here does not match the results seen in other sections.

*6. K-Nearest Neighbors (KNN)*

The last model of interest that was looked at was the K-Nearest Neighbor (KNN) method. Shown below in Fig. 5.6, the testing and training scores were plotted for various values of K. Thus, it k=4 was determined to be the most accurate as the variance between the training and testing scores was minimal.

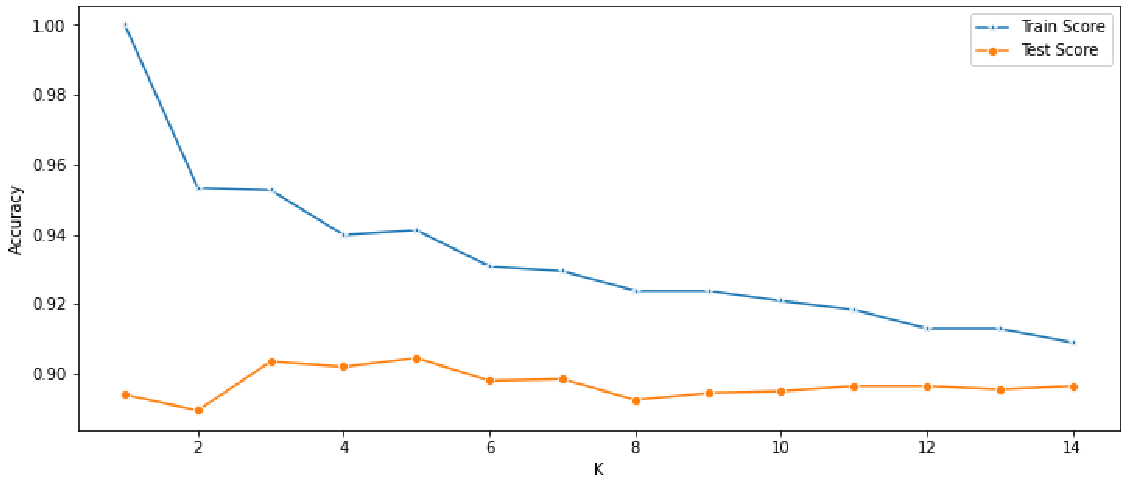
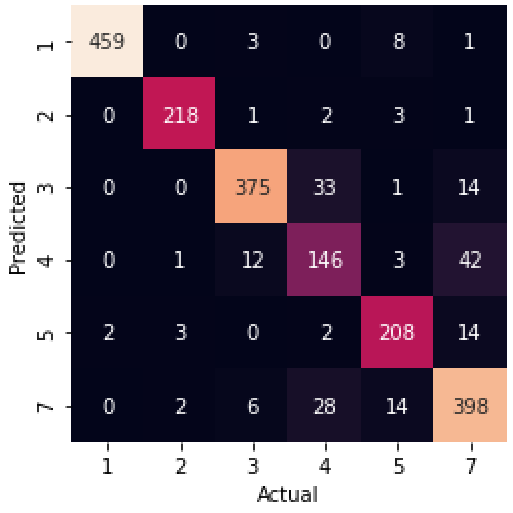


**Figure 5.6. Testing and Training Scores for K-Nearest Neighbors**

Then, with a value of k=4, the relevant confusion matrix shown below in Fig. 5.7 was able to be generated.



| Predicted \ Actual | 1 | 2 | 3 | 4 | 5 | 7 |
|---|---|---|---|---|---|---|
| 1 | 459 | 0 | 3 | 0 | 8 | 1 |
| 2 | 0 | 218 | 1 | 2 | 3 | 1 |
| 3 | 0 | 0 | 375 | 33 | 1 | 14 |
| 4 | 0 | 1 | 12 | 146 | 3 | 42 |
| 5 | 2 | 3 | 0 | 2 | 208 | 14 |
| 7 | 0 | 2 | 6 | 28 | 14 | 398 |

**Figure 5.7. KNN Confusion Matrix**

**Table 5.6. KNN Metrics**

| Metrics | Values |
|---|---|
| Average Execution Time | 11.73 s |
| Accuracy | 90.2% |

In terms of accuracy, the KNN method proved to be the best among the six methods tested. With an accuracy of 90.2%, it was 2.7% more accurate than the second-best method, polynomial SVM. Nevertheless, upon comparing the two confusion matrices, the key area of improvement that KNN succeeds in is its ability to classify the grey soil accurately. As mentioned, polynomial SVM only predicted damp grey soil with a 52% accuracy rate which is far less than the accuracy rate for the other individual labels. Here, KNN produced an accuracy rate for damp grey soil of 70%, which though only moderately accurate, is far better than any of the five methods. Visually, the confusion matrix demonstrates the model's precision by having the diagonal entries to be colored the lightest -- meaning that the labels that were predicted the most were also the same as the actual label. However, at the cost of having a high accuracy rate, the KNN method has the slowest execution time among the previous methods. Compared to the SVM methods which had execution times within the range of milliseconds, the KNN method was magnitudes slower. Thus, for comparably smaller datasets as the one used in this paper, a time of 11.73 seconds is not entirely damaging; however, if these results were to be extrapolated to much larger datasets for images covering a wider landscape, the long execution time of KNN would become a more detrimental factor.

## *7. Summary and Model Evaluation*

After analyzing each individual model's ability to classify the dataset, this section will now compare the accuracy metrics of each method against each other to determine the optimal method. Recall that there were three types of metrics involved: response time, accuracy, and f-score. Since the f-score was calculated on an individual basis for each label, the results of the f-score values have been shown below in Table 5.7. There, the method which performed with the greatest f-score for each label has been highlighted in green.

**Table 5.7. F-Score for Various Methods**

| Label | Logistic Regression | Linear SVM | Polynomial SVM | RBF SVM | Naïve Bayes | KNN |
|---|---|---|---|---|---|---|
| 1 (Red Soil) | 0.98 | 0.90 | 0.99 | 0.99 | 0.98 | 0.98 |
| 2 (Cotton Crop) | 0.94 | 0.91 | 0.97 | 0.97 | 0.93 | 0.97 |
| 3 (Grey Soil) | 0.89 | 0.87 | 0.91 | 0.91 | 0.73 | 0.91 |
| 4 (Damp Grey Soil) | 0.35 | 0.44 | 0.52 | 0.52 | 0.08 | 0.70 |
| 5 (Soil w/ Vegetation Stubble) | 0.80 | 0.52 | 0.86 | 0.86 | 0.73 | 0.89 |
| 7 (Very Damp Grey Soil) | 0.80 | 0.75 | 0.84 | 0.84 | 0.72 | 0.87 |

As expected from the previous sections, the methods that were most precise were polynomial SVM, RBF SVM, and K-NN. Moreover, the K-Nearest Neighbor method had the most precise model in each case (except for red soil which was 1% less accurate). In particular, KNN had the ability to accurately classify the damp grey soil by a larger margin than any other label. Thus, in terms of accuracy, the KNN model is the most optimal not only for its high general accuracy, but for its ability to correctly identify the grey soil types. This topic will be explained in further detail in the next section, but it is worth noting that though KNN is 3% more accurate, the vast improvement in grey soil classification affirms KNN's high accuracy.

However, if the results of the models are to be considered in a real-world environment, it will be of more importance to consider execution time in the selection of an optimal model. Though KNN was more accurate than any of the models, it also had the slowest running time of 11.73 seconds. Though the classification algorithms were being run on a laptop, the issue of time may still be a problem on higher-quality machines. Thus, for a general optimal model, the RBF SVM method will likely be most efficient. For instance, real world models may have a much larger data set compromising multiple images; so for general classification purposes, RBF SVM would be helpful to get an accurate approximation of the soil types. However, for the instance where a user may want to get the most accurate analysis of the region, then KNN would be best as execution time is not necessarily of great import.

## B. Landsat Image Data Analysis

From the above analysis we have determined that the SVM (RBF kernel) and the K-Nearest Neighbor are the most optimal classification models for the given Landsat dataset. We can now begin analyzing the image data through these models.

First, to get a better idea of the predictions made by the SVM model, a bar plot has been generated,

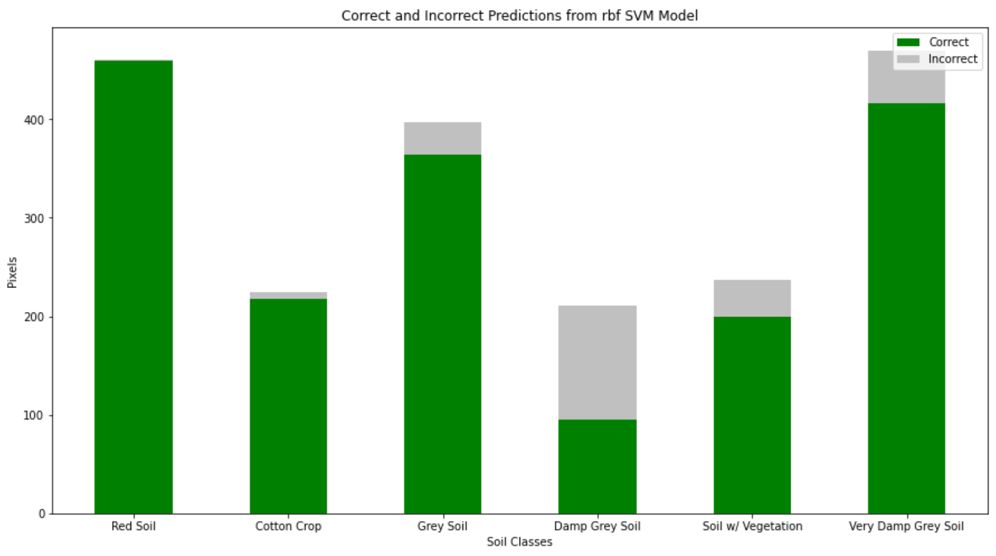


**Figure 5.8. Soil Predictions for SVM (RBF Kernel)**

Figure 5.8 allows us to visualize the interesting results that were provided by this classification algorithm. It has already been noted several times in the analysis that most models had difficulties in differentiating between the three grey soil categories. The plot above helps reinforce this evaluation, but it seems that the SVM method had the most trouble with the "Damp Grey Soil" category. Surprisingly, a majority of the predictions for this soil class were incorrect. The other grey soil types experienced inaccuracy as well, but damp grey soil was by far the most problematic. It can also be noted that the "soil with vegetation" class had a surprisingly high level of incorrect predictions.

In comparison, the same bar plot for the K-Nearest Neighbor model can be seen below,

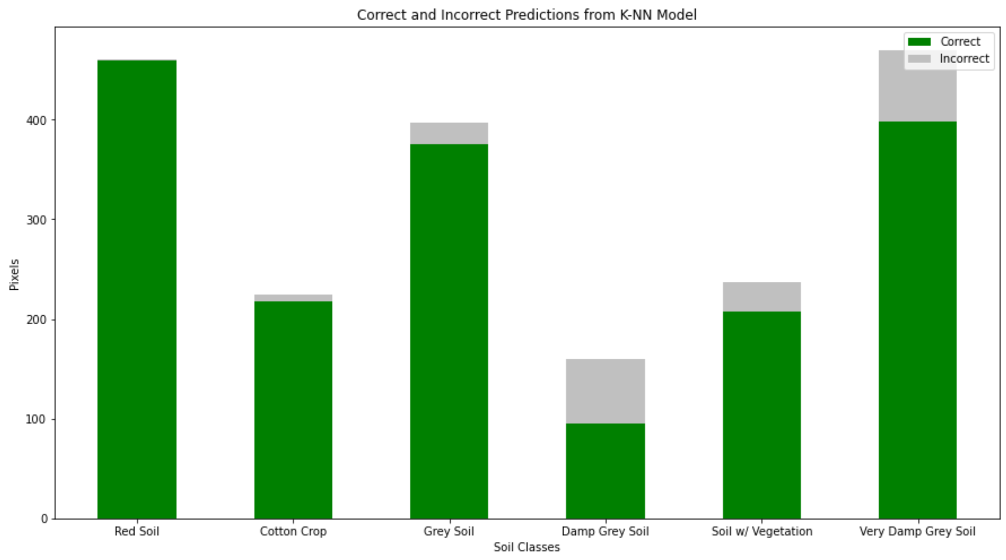


**Figure 5.9. Soil Predictions for K-NN**

As expected, the grey soil predictions caused the highest amount of problems for the K-NN model. While the incorrect predictions were clearly less than the amount found in the SVM model, this model still suffers from this pertinent issue. Additionally, the "Soil w/ Vegetation" category still remained the one non-grey soil class that had a non-insignificant amount of incorrect predictions.

It seems that even when the classification method was optimized as much as possible, the grey soil categories still generated significant errors for both of the optimal models. If we reproduce the original heat map confusion matrices for both of these methods and remove all non-grey soil types, the following figure is generated.

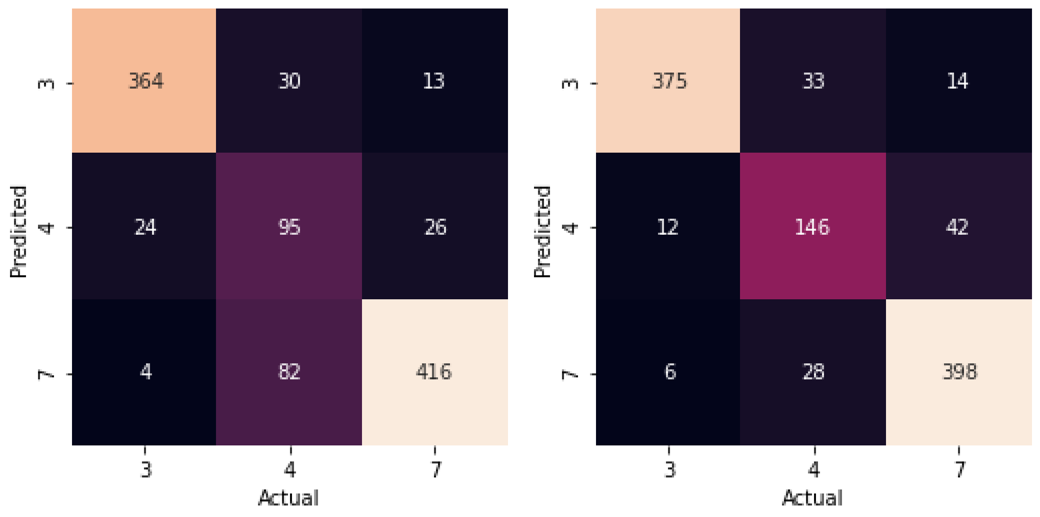


**Figure 5.10. Confusion Matrices for Grey Soil Categories. From Left: a) SVM, b) K-NN**

Figure 5.10 is extremely fascinating because it allows us to gain a better understanding of where the issues lie within the grey soil dilemma. It seems that for the simple “Grey Soil” (3) category, both models did a similar job in the accuracy of their predictions. Conversely, the SVM model had significantly more difficulties with the “Damp Grey Soil” class. There is barely a clear winner in predictions as both category 4 and 7 have 95 and 82 cases respectively. While the SVM is underperforming so far, it manages to beat the K-NN in the classification of “Very Damp Grey Soil”.

Overall, both methods have high accuracies, but the inclusion of three similar grey soil categories hurt both of the models. It is clear than more training data is required for categories 3 through 7.

## VI. Conclusion

The overall objective of the project was to analyze a given dataset of imaged topological data and classify a given pixel with the likely type of soil that is represented. Moreover, this paper explored finding an optimal method that was accurate and efficient in its ability to classify the data. For the six methods that were involved, a number of analytical techniques were made to best assess the quality of each method’s performance. The following points summarize the conclusions and results made by the analysis and discussion in the previous sections:

1. The SVM (Radial Basis Function Kernel) and K-NN models were the most optimal methods for pixel classification
2. Errors mainly arose from the similarity of grey soil classes; therefore, the separation of these three groups should be reconsidered.

For point (1), it was found that the RBF SVM method and KNN models were the best methods of classifying pixel data. The specific choice of which method would be used would also have to be decided on a case-by-case basis. This would have to be done considering that KNN had the most accurate model but had the slowest execution time. Overall, though, RBF SVM had a quality accuracy rate and fast response rate. Thus, in a real world environment, if the user is scanning a large area of land and wants a general approximation of the type of soil, then RBF SVM would be best. On the other hand, if the user wanted a highly accurate mockup of the topological composition, then KNN would likely be used as response time would be a less important metric.

For point (2), as has been mentioned throughout this report, the “Grey”, “Damp Gray” and “Very Damp Gray” soil categories caused the most difficulties for all of the classification models that were used. Due to these soil types being so similar to each other, it is easy to see why the multispectral pixel data would be close enough to trick the models into incorrect predictions. From this discovery, it is highly recommended to determine the correct fix for this issue, as having half of possible classifications being sources of trouble is not an optimal situation. One possible solution would be to introduce more training data for all of the models -- especially data with an emphasis on the grey soil categories. Not only will this training data help differentiate these similar soil types, it will also benefit the accuracy of the system as a whole. Additionally, it can be recommended to reformulate the data and/or overall problem in order to mitigate this issue. For example, if “Damp Grey” and “Very Damp Gray” were to be combined, then there would be less incorrect predictions for the classification models. Obviously this depends on the mission parameters and objectives, but a restructuring must be considered in order to improve classification from the image data.

## ACKNOWLEDGEMENTS

| | |
|---|---|
| Prof. Raju, | Thank you for building and teaching a fantastic course. |
| Suyash Ail, | Thank you being a great TA and answering our questions |

## REFERENCES

[1] "What is Remote Sensing?," *EARTHDATA*. [Online]. Available: https://earthdata.nasa.gov/learn/backgrounders/remote-sensing [Accessed: 03-Dec-2020].

[2] A. Srinivasan, "Statlog (Landsat Satellite) Data Set," *Machine Learning Repository*. [Online]. Available:https://archive.ics.uci.edu/ml/datasets/Statlog+%28Landsat+Satellite%29. [Accessed: 03-Dec-2020].

[3] "Remote Sensing Market Map: 20 Remote Sensing Startups and the Varied Data That Fuels Them," *AgFunderNews*. [Online]. Available: https://agfundernews.com/remote-sensing-market-map.html. [Accessed: 03-Dec-2020].

[4] Weiss, M, and F Jacob. "Remote Sensing for Agricultural Applications: A Meta-Review." Remote Sensing of Environment, vol. 236, Jan. 2020, doi:https://doi.org/10.1016/j.rse.2019.111402

[5] Ma, Lei, et al. "Deep Learning in Remote Sensing Applications: A Meta-Analysis and Review." *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 152, June 2019, doi:https://doi.org/10.1016/j.isprsjprs.2019.04.015.

[6] Saiz-Rubio, Verónica and Rovira-Más, Francisco. "From Smart Farming towards Agriculture 5.0: A Review on Crop Data Management." Agronomy, vol. 10, Feb. 2020, doi:doi.org/10.3390/agronomy10020207

[7] D. Jurafsky and J. H. Martin, "Chapter 5: Logistic Regression," in *Speech and language processing*, Harlow: Pearson, 2014.

[8] A. Zisserman. (2020). Lecture 2: The SVM classifier [PowerPoint Slides]. Available: https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

[9] Z. Zhang. "Introduction to Machine Learning: K-nearest neighbors." Annals of Translational Medicine, vol. 11, May. 2016, doi: 10.21037/atm.2016.03.37

[10] H. V. Raju. (2020). ME597: Artificial Intelligence in Thermal Systems [PowerPoint Slides].

## APPENDIX

```
#Load libraries
import pandas as pd
from pandas import read_csv
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split  #importing the
train_test_split function from the sklearn library
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn import svm
from sklearn.naive_bayes import MultinomialNB
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier

#Load the test and train datasets
train = pd.read_excel('train.xlsx')
test = pd.read_excel('test.xlsx')

values_train = train.values
values_test = test.values

train_x, train_y = values_train[:, :-1], values_train[:, -1]
test_x, test_y = values_test[:, :-1], values_test[:, -1]

%%time
# Generate a Logistic  regression model
from sklearn import preprocessing

model =
LogisticRegression(penalty="l2",C=1.0,fit_intercept=True,max_iter=3000)
model.fit(train_x, train_y)
y_hat = model.predict(test_x)

#print(y_hat)
#print(test_y)

# Print Accuracy
score = accuracy_score(test_y, y_hat)
print("Logistic Regression Accuracy: ", score)

# Generate confusion matrix and heat map for Logistic Regression
mat = confusion_matrix(test_y, y_hat)
x_axis_labels = [1,2,3,4,5,7] # labels for x-axis
y_axis_labels = [1,2,3,4,5,7] # labels for y-axis
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');
print(classification_report(test_y,y_hat))

%%time
# Linear Kernal
```

```
model_lin = svm.SVC(kernel='linear',degree=64,C=1,max_iter=500)
model_lin.fit(train_x,train_y)

#evaluate model accuracy
y_hat_lin = model_lin.predict(test_x)
score = accuracy_score(test_y, y_hat_lin)
print("Linear SVM Accuracy: ", score)

#Build a confusion matrix and visualize as a heat map
mat = confusion_matrix(test_y, y_hat_lin)
x_axis_labels = [1,2,3,4,5,7] # labels for x-axis
y_axis_labels = [1,2,3,4,5,7] # labels for y-axis
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');
print(classification_report(test_y,y_hat_lin))

%%time
# Polynomial Kernal

model_poly = svm.SVC(kernel='poly',degree=2,C=1,max_iter=1000)
model_poly.fit(train_x,train_y)

#evaluate model accuracy
y_hat_poly = model_poly.predict(test_x)
score = accuracy_score(test_y, y_hat_poly)
print("Polynomial SVM Accuracy: ", score)

#Build a confusion matrix and visualize as a heat map
mat = confusion_matrix(test_y, y_hat_poly)
x_axis_labels = [1,2,3,4,5,7] # labels for x-axis
y_axis_labels = [1,2,3,4,5,7] # labels for y-axis
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');
print(classification_report(test_y,y_hat_poly))

%%time
# RBF Kernal

model_rbf = svm.SVC(kernel='rbf',degree=2,C=1,max_iter=1000)
model_rbf.fit(train_x,train_y)

#evaluate model accuracy
y_hat_rbf = model_poly.predict(test_x)
score = accuracy_score(test_y, y_hat_rbf)
print("RFB SVM Accuracy: ", score)

mat_rbf = confusion_matrix(test_y, y_hat_rbf)
x_axis_labels = [1,2,3,4,5,7] # labels for x-axis
y_axis_labels = [1,2,3,4,5,7] # labels for y-axis
sns.heatmap(mat_rbf.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
```

```
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');
print(classification_report(test_y,y_hat_rbf))

%%time
# Generate a Naive Bayes model

model=MultinomialNB()

model.fit(train_x, train_y)
labels = model.predict(test_x)

#predict the accuracy of the model on test data
score = accuracy_score(test_y, labels)
print("Naive Bayes Accuracy: ", score)

#build a confusion matrix to visualize the performance of the model using
seaborn
mat = confusion_matrix(test_y, labels)
x_axis_labels = [1,2,3,4,5,7]
y_axis_labels = [1,2,3,4,5,7]
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');
#Print the Classification Report
from sklearn.metrics import classification_report
print(classification_report(test_y,labels))

%%time

# manually try different k values to get the optimal value.

test_scores = []
train_scores = []
for i in range(1,15):
    knn = KNeighborsClassifier(i)
    knn.fit(train_x,train_y)
    pred_y_train=knn.predict(train_x)
    train_accuracy=accuracy_score(pred_y_train,train_y)
    train_scores.append(train_accuracy)
    pred_y_test=knn.predict(test_x)
    test_accuracy=accuracy_score(pred_y_test,test_y)
    test_scores.append(test_accuracy)

plt.figure(figsize=(12,5))
p = sns.lineplot(range(1,15),train_scores,marker='*',label='Train Score')
p = sns.lineplot(range(1,15),test_scores,marker='o',label='Test Score')
plt.xlabel("K")
plt.ylabel("Accuracy")

%%time

# Use highest accuracy K for final K-nearest model
print("K value used:", test_scores.index(max(test_scores)))
```

```
knn1 = KNeighborsClassifier(test_scores.index(max(test_scores)))
knn1.fit(train_x,train_y)
y_pred = knn1.predict(test_x)
score = accuracy_score(y_pred, test_y)
print("K-nearest Neighbours Accuracy: ", score)

#build a confusion matrix to visualize the performance of the model using
seaborn
mat_knn = confusion_matrix(test_y, y_pred)
x_axis_labels = [1,2,3,4,5,7] # labels for x-axis
y_axis_labels = [1,2,3,4,5,7] # labels for y-axis
sns.heatmap(mat_knn.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');
print(classification_report(test_y,y_pred))

from matplotlib.pyplot import figure

# Plot bar chart of predicted soil classes
predict_1_correct = mat_rbf[0,0]
predict_1_incorrect = sum(mat_rbf[0,1:6])
predict_2_correct = mat_rbf[1,1]
predict_2_incorrect = mat_rbf[1,0]+ sum(mat_rbf[1,2:6])
predict_3_correct = mat_rbf[2,2]
predict_3_incorrect = sum(mat_rbf[2,0:2]) + sum(mat_rbf[2,3:6])
predict_4_correct = mat_rbf[3,3]
predict_4_incorrect = sum(mat_rbf[3,0:3]) + sum(mat_rbf[3,4:6])
predict_5_correct = mat_rbf[4,4]
predict_5_incorrect = mat_rbf[4,5]+ sum(mat_rbf[4,0:4])
predict_7_correct = mat_rbf[5,5]
predict_7_incorrect = sum(mat_rbf[5,0:5])

predict_correct = [predict_1_correct, predict_2_correct, predict_3_correct,
predict_4_correct, predict_5_correct, predict_7_correct]
predict_incorrect= [predict_1_incorrect, predict_2_incorrect,
predict_3_incorrect, predict_4_incorrect, predict_5_incorrect,
predict_7_incorrect]
#print(predict_incorrect)
#print(predict_correct)

countries = ['Red Soil', 'Cotton Crop', 'Grey Soil', 'Damp Grey Soil', 'Soil
w/ Vegetation', 'Very Damp Grey Soil']
ind = [1,2,3,4,5,6]
plt.figure(figsize=(15,8))

plt.bar(ind, predict_correct, width=0.5, label='Correct', color='green')
plt.bar(ind, predict_incorrect, width=0.5, label='Incorrect', color='silver',
bottom=predict_correct)

plt.xticks(ind, countries)
plt.ylabel("Pixels")
plt.xlabel("Soil Classes")
plt.legend(loc="upper right")
plt.title("Correct and Incorrect Predictions from rbf SVM Model")
plt.show()
```

```
# Grey Soil Analysis For Rbf Model
grey_rbf = np.delete(np.delete(mat_rbf, [0,1,4], 0), [0,1,4], 1)

x_axis_labels = [3,4,7] # labels for x-axis
y_axis_labels = [3,4,7] # labels for y-axis
sns.heatmap(grey_rbf.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');

# Plot bar chart of predicted soil classes
predict_1_correct = mat_knn[0,0]
predict_1_incorrect = sum(mat_rbf[0,1:6])
predict_2_correct = mat_knn[1,1]
predict_2_incorrect = mat_knn[1,0]+ sum(mat_knn[1,2:6])
predict_3_correct = mat_knn[2,2]
predict_3_incorrect = sum(mat_knn[2,0:2]) + sum(mat_knn[2,3:6])
predict_4_correct = mat_rbf[3,3]
predict_4_incorrect = sum(mat_knn[3,0:3]) + sum(mat_knn[3,4:6])
predict_5_correct = mat_knn[4,4]
predict_5_incorrect = mat_knn[4,5]+ sum(mat_knn[4,0:4])
predict_7_correct = mat_knn[5,5]
predict_7_incorrect = sum(mat_knn[5,0:5])

predict_correct = [predict_1_correct, predict_2_correct, predict_3_correct,
predict_4_correct, predict_5_correct, predict_7_correct]
predict_incorrect= [predict_1_incorrect, predict_2_incorrect,
predict_3_incorrect, predict_4_incorrect, predict_5_incorrect,
predict_7_incorrect]
#print(predict_incorrect)
#print(predict_correct)

countries = ['Red Soil', 'Cotton Crop', 'Grey Soil', 'Damp Grey Soil', 'Soil
w/ Vegetation', 'Very Damp Grey Soil']
ind = [1,2,3,4,5,6]
plt.figure(figsize=(15,8))

plt.bar(ind, predict_correct, width=0.5, label='Correct', color='green')
plt.bar(ind, predict_incorrect, width=0.5, label='Incorrect', color='silver',
bottom=predict_correct)

plt.xticks(ind, countries)
plt.ylabel("Pixels")
plt.xlabel("Soil Classes")
plt.legend(loc="upper right")
plt.title("Correct and Incorrect Predictions from K-NN Model")
plt.show()

# Grey Soil Analysis For K-NN Model
grey_knn = np.delete(np.delete(mat_knn, [0,1,4], 0), [0,1,4], 1)

x_axis_labels = [3,4,7] # labels for x-axis
y_axis_labels = [3,4,7] # labels for y-axis
sns.heatmap(grey_knn.T, square=True, annot=True, fmt='d', cbar=False,
xticklabels=x_axis_labels, yticklabels=y_axis_labels
```

```
    )
plt.xlabel('Actual')
plt.ylabel('Predicted');
```