Abstract

The aims of this project to detect different objects in the photo, like water, cows, and grass by using different models. In addition to compare the accuracy of these models with different parameters.

Figure 1: The original image

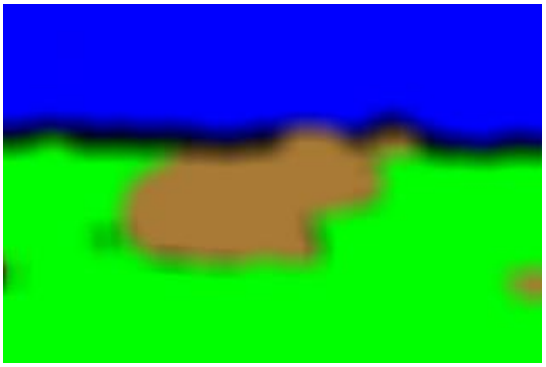

Figure 2: The image after processing

Contents

Figures

Introduction

Task Definition

In this project we are trying segment images into multiple class based to 4 labels ,Water, Grass, Cow and other things, this problem can be formalized as classification problem by having some training images, and each of them are segment into four labels, our goal is to create a models that can label any other images with the same type of data contain water and cross and cows, if we want to test more complicated images we want to add more training images that mirror the images we want to label.

Models

In this project we use the following models:

- KNN
- Decision Tree
- Logistic Regression
- Random Forest as ensemble bagging algorithm
- X Gradient as ensemble boosting algorithm

Metrics

- For KNN Model we use Minkowski as distance metric

Cross Validation

For all Models Grid Search Cross Validation is Used For tuning hyper parameters for all models.

Technical Details

KNN Algorithm

Quick overview

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

Grid Search Cross Validation

We used the cross validation using grid search technique which have a grid of hyper parameters as input and try all the combinations of this of these parameters and give the best combinations of the as output.

For KNN we try this grid of hyper parameters [1,3,5], the values chosen randomly and the best K for KNN model was 1.

Minkowski Distance Metric

For distance metric we use the Minkowski Distance as distance metric, Minkowski distance is a distance similarity measurement between two points in the normed vector space
(N dimensional real space) and is a generalization of the Euclidean distance and the Manhattan distance.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

The value of P in Minkowski Distance is set based on experimentation. The usual value of P is either 2, 3 or 4 for most problems, in our problem we set the value of P to 2(The default value in sklearn API).

Decision Tree

Quick overview

A Decision tree is a flowchart like tree structure, where each internal 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.

Grid Search Cross Validation

In this project we used Grid Search Cross Validation to determine the best depth for the tree, we used the random values [50,120,90] in cross validation and founded that the 120 is the best.

Logistic Regression

Quick overview

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

Grid Search Cross Validation

We used Grid search Cross Validation to determine the best value for the parameter C of the logistic model, we pass 4 random values [0.1,0.2,0.3,0.4] to the cross validation and we get that the best value is 0.1.

The parameter C is the regularizer for logistic regression, regularization in logistic regression used to prevent the over fitting of the model and make it more general for unseen data, which mean that it will give a good result for unseen data sets.

Random Forest

Quick overview

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Random forest is one of the algorithms that use the bagging ensemble method.

Grid search Cross Validation

In random forest we used Grid Search Cross Validation to determine the best value for the parameters n_estimators and max_depth.

n_estimators mean the number of decision trees that the model will use, we try the values [50,100,200,300].

Max_depth mean the depth for each estimator trees, we use the values [2,50,100,300].

We got that the n_esttimators 300 and max_depth 50 gave the best result.

XGradiant Boosting

Quick overview

The main idea behind this gradient boosting is to build models sequentially and these subsequent models try to reduce the errors of the previous model, This is done by building a new model on the errors or residuals of the previous model, the main different between the AdaBoost and the gradient boost that the Ada Boost increase the wights of the week learners (e.g.. Repeat the data that classified wrong) and the Gradient boost built the new model sequentially based on gradient decent, and the new model that created minimize the error.

XGboost (Extreme Gradient Boosting)

XGBoost is a more regularized form of Gradient Boosting, it uses advanced regularization (L1 & L2), which improves model generalization capabilities. it delivers high performance as compared to Gradient Boosting. Its training is very fast and can be parallelized across clusters.

Grid search Cross Validation

In XG Boosting we use Grid search Cross Validation to determine the best value for the parameters n_estimators, learning rate.

n_estimators belong to the number of trees to build, we use the values [200,300,400] in grid search.

Learning rate is same as neural network we use learning rate Alpha to update the weights, to reach the minimum error, here the learning rate do the same thing when creating a new model the learning rate used in updating the model output to reach the minimum error for the date we have, we used the values [0.1,0.2,0.3] in the grid search.

We got that 300 n_estimator and learning rate 0.2 was the best values for this model.

Feature Vector Extraction

In our color histogram program, we are using a tuple, (channel_id, c), as variables. The first time through the loop, the channel_id variable takes the value 0, referring to the position of the red colour channel, and the c variable contains the string "red". The second time through the loop the values are the green channels position and "green", and the third time they are the blue channel position and "blue".

Inside the for loop, our code looks much like it did for the grayscale example. We calculate the histogram for the current channel with the histogram:

bin_edges = np.histogram(image[:, :, channel_id], bins=256, range=(0, 256)) function call, and then add a histogram line of the correct color to the plot with:

plt.plot(bin_edges[0:-1], histogram, color=c) function call.

Note the use of our loop variables, channel_id and c. Finally, we label our axes and display the histogram, shown here:

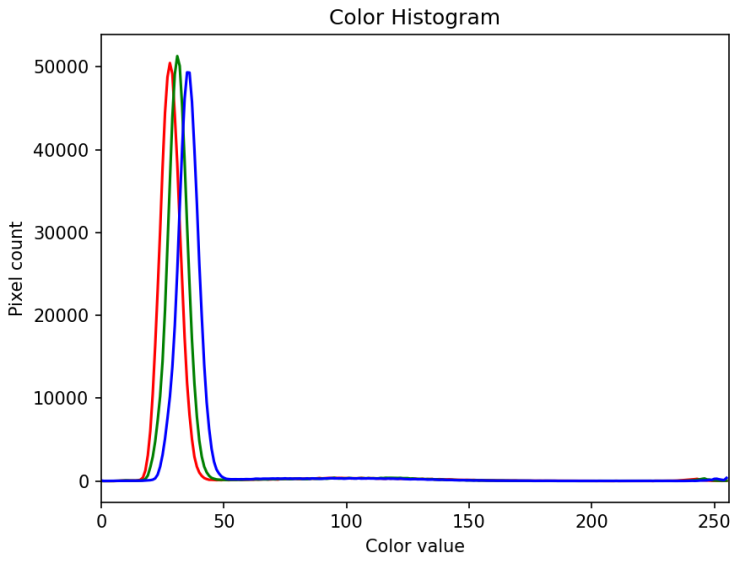

Figure 3: Feature Vector Extraction

For each of the RGB channels, create a histogram with 64 bins, where bin i contains the number of pixels with intensity value in the range [i*4, (i+1)*4), and 0 <= i < 64.

```
features_vector = []

for i in range(number_of_colors):
    count[COLORS[i]] = [sum(count[COLORS[i]][j * 4: (j + 1) * 4])
                        for j in range(64)]
    features_vector.extend(count[COLORS[i]])

return features_vector
```

Figure 4: Convert 256 pins to 64 pins for each color in RGB

Training Data

We got a 1500 patch for each class from each image as maximum, each patch (17, 17) represents a sample in features vector and link the target with it.

Testing Data

We got all patches from the image and link the targets with it.

Experiments and Results

We used different models with different parameters for detect the cow, water, and grass in the image.

Figure 5: The testing image

Figure 6: The target of the testing image

KNN

In the KNN algorithm the accuracy for training was 100% but in testing Data the accuracy was 90%, the photo has noise distributed over all the image, the main cause that the decision is by a patch, and we do not have any type of generalization (there is no learning) because we find where the labels for k neighbors for the patch and then do the majority voting.

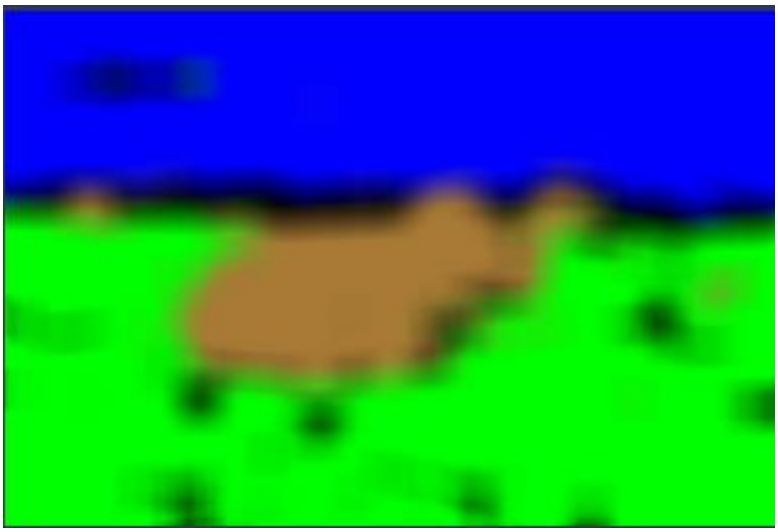

Figure 7: KNN Result

We tried a lot to improve the performance for KNN model by make it weighted KNN and change the distance metric but the result still the same.

Decision Tree

In the DT the accuracy for training was 100% and the Accuracy for testing was 82.6%, it's clear that we have overfitting in the training data.

The Decision tree does not give a good result for the testing data, we try to improve the efficacy of it by using pruning, but we don't have a much different, the training Accuracy down to 86.6% and the testing improve to 83.7%, so to get a good improvement we go to ensemble methods like Random Forest(bagging) and Gradient Boosting, the results for these two models will be discussed later.

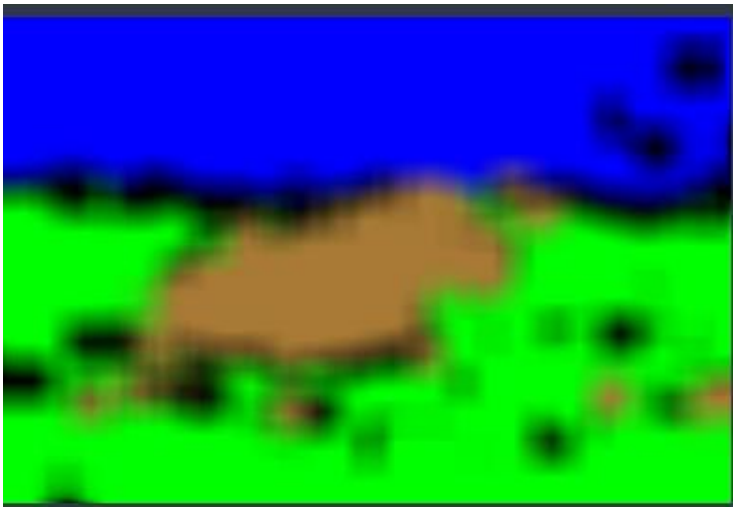

Figure 8: Decision Tree Result

Logistic Regression

In the Logistic regression the accuracy for training was 90% and the accuracy for testing was 87.3%, the result was good, still has noise but the noise is concentrated in one place, the result was better than decision tree.

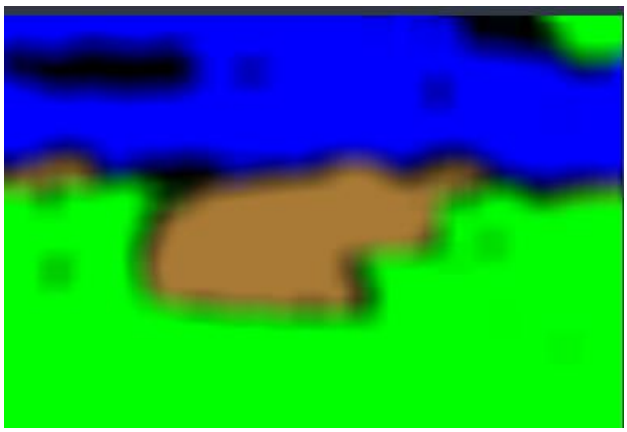

Figure 9: Logistic Regression Result

Although that the accuracy is less than KNN and has more noise but from result we can see that the logistic regression is more generalized.

Random Forest

Mainly Random forest used to improve the performance of the Decision Tree, we use multiple trees and use majority voting for the decision for the result from all trees, and we notice a good improvement for the results, the Accuracy in the training was 100% and the accuracy in testing is 93%,after tuning the hyper parameters the accuracy improved to 93.7% and the result images was very good, and the noise is a little bit.

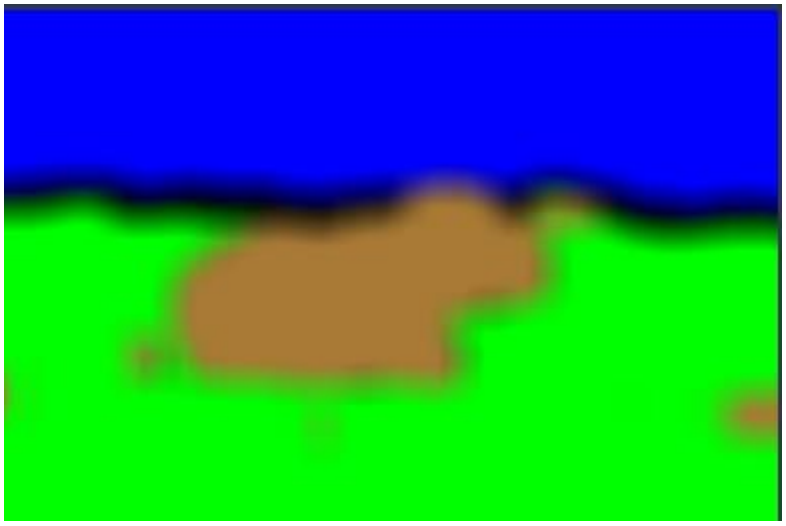

Figure 10: Random Forest Result

XGBoosting

As mentioned in DT we go to use ensemble methods to improve the decision Tree, XGBoost was the best model and it's the king 👑 👑 .

Without tuning the hyperparameters the accuracy in the training set was 99% and the accuracy in the testing set was 94%, the result is good and the noise is less than the random forest result, after tunning the hyperparameters, with multiple runs the accuracy for testing set was in Range (94.4-94.9), the random forest better than the XGBoost in speed, But if we use the Hyperparameter Tree_method in XGBoost and set it to gpu_hist it allow the algorithm to use the GPU in your personal Machine and you will have the result in time like random forest.

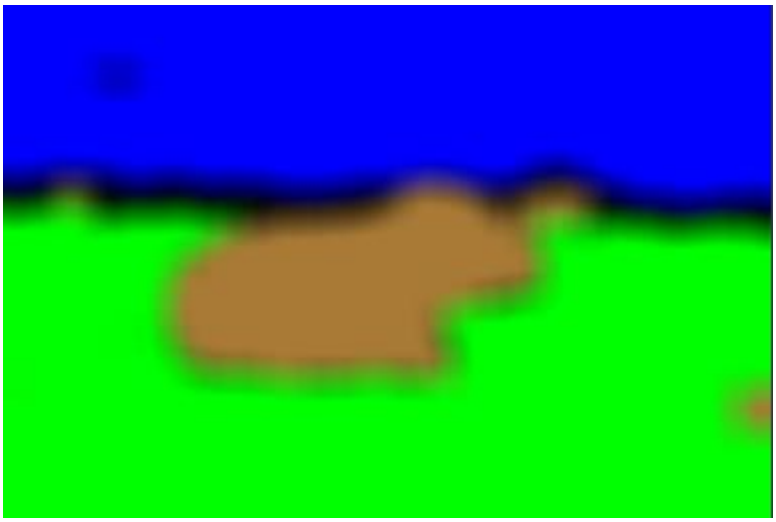

Figure 11: XGBoosting Result

Conclusion

In this project we managed to get good results, the models like KNN, Decision Tree, Logistic regression can’t give the desired results however you change in the hyperparameters, the solution is to go to models have advanced architectures can give a better results like random forest and boosting.

We made a practical application of some of the principles that we learned in the theoretical course and were able to understand them more, and we learned about new things that we did not take in the course, we read many articles about them, and we learned to use many software libraries that we did not hear about before, the project It was very useful and opened up new horizons for us in machine learning, and we hope to develop ourselves well in this field.

References

- https://towardsdatascience.com/3-techniques-to-avoid-overfitting-of-decision-trees-1e7d3d985a09#:~:text=Post%2DPruning%3A,prevent%20the%20model%20from%20overfitting.

- https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

- https://xgboost.readthedocs.io/en/stable/parameter.html

- https://towardsdatascience.com/why-do-we-set-a-random-state-in-machine-learning-models-bb2dc68d8431

- https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20forest%20is%20a%20Supervised,average%20in%20case%20of%20regression.

- https://www.knime.com/blog/regularization-for-logistic-regression-l1-l2-gauss-or-laplace#:~:text=%E2%80%9CRegularization%20is%20any%20modification%20we,from%20overfitting%20the%20training%20dataset.

- https://www.knime.com/blog/regularization-for-logistic-regression-l1-l2-gauss-or-laplace#:~:text=%E2%80%9CRegularization%20is%20any%20modification%20we,from%20overfitting%20the%20training%20dataset.