IoT Data Analytics

University of Salerno

Department of Computer Science

Forest Covertype

Tutor
Stefano Cirillo

Teacher
Giuseppe Polese
Genoveffa Tortora

Students
Valerio Passamano
0522500919
Antonio Sicuranza
0522500987

Contents

Chapter 1

Introduction

1.1 Goal of the project

The objective of the project, as part of the IoT Data Analytics exam, is to predict the forest covertype with the sole use of cartographic variables.

To do this, the "Covertype Data Set" dataset was used, available at the following link:

`https://archive.ics.uci.edu/ml/datasets/Covertype`

The dataset, although static, was used to simulate data streams for training and predictor evaluation.

1.2 State of art

The Forest Cover Type problem aims to look for Machine Learning models as accurate as possible. In order to work with the dataset it is important to have a basic understanding of how the data was collected to evaluate its accuracy and veracity. A first aid in understanding can be given by the study entitled: "Classifying and mapping forest cover types using IKONOS imagery in the NorthEastern United States "[1].

The advancement of higher spatial resolution satellite sensors have historically allowed for increases in the accuracies of mapping forest cover types. These accuracies have been further increased by image processing techniques such as hybrid forms of the supervised and unsupervised classification methods. However, since the launch of very high resolution (VHR) sensors ($\leq 4m$) such as IKONOS, traditional per-pixel automated classification has not always worked well. While increases in spatial resolution create increased amounts of informational detail, this also creates higher within class spectral variability, potentially causing lower classification accuracies when solely using per-pixel classification techniques based upon spectral comparisons. By incorporating other image processing techniques such as texture into automated classification, the increased within class spectral variability inherent to very high resolution images may increase our ability to discern relatively homogeneous clusters of pixels.

It should then be understood how Machine Learning techniques can help us in the search for a solution as precise and performing as possible, in this the study "Comparative accuracies of artificial neural gives us a hand. networks and discriminant analysis in predicting forest cover types from cartographic variables " [2] that examined the ability of an ANN model to predict forest cover type classes in forested areas that have experienced relatively little direct human management activities in the recent past.

According to this study Compared to statistical models, artificial neural networks (ANNs) represent a relatively new approach to developing predictive models.

Artificial neural networks are 'computing devices that use design principles similar to the design of the information-processing system of the human brain " (Bharath and Drosen, 1993, p. xvii).

Several recent textbooks describe the mechanics of ANNs (Hertz et al., 1991; Haykin, 1994; Masters, 1994; Bishop, 1995; Ripley, 1996). Recent publications involving artificial neural networks being applied to natural resources topics includes: modeling complex biophysical interactions for resource planning applications (Gimblett and Ball, 1995); generating terrain textures from a digital elevation model and remotely sensed data (Alvarez, 1995); modeling individual tree survival probabilities (Guan and Gertner, 1995); and Harvey and Dean (1996), who used geographic information systems (GIS) in developing computer-aided visualization of proposed road networks. Recent comparisons in which ANNs performed favorably against conventional statistical approaches include Reibnegger et al. (1991), Patuwo et al. (1993), Yoon et al. (1993), Marzban and Stumpf (1996), Paruelo and Tomasel (1996), Pattie and Haas (1996), and Marzban et al. (1997).

However, artificial neural networks do not always outperform traditional predictive models. For example, Jan (1997) found a traditional maximum-likelihood classifier outperformed artificial neural network models when classifying remotely sensed crop data. Also, using their best artificial neural network model, Vega-Garcia et al. (1996) obtained only a slight improvement in predicting human-caused wildfire occurrences as compared to their best logit model.

1.3 The Dataset

The dataset used was developed by Colorado State University as part of the "Remote Sensing and GIS Program".

The data refer to 30x30 meters large clods of land in the "Roosevelt National Forest of Northern Colorado" for a total of 581012 tuples.

Each tuple is made up of 54 attributes, which we are now going to list:

Name	Data Type	Measurement	Description
Elevation	quantitative	meters	Elevation in meters
Aspect	quantitative	azimuth	Aspect in degrees azimuth
Slope	quantitative	degrees	Slope in degrees
Horizontal_Distance_To_Hydrology	quantitative	meters	Horz Dist to nearest surface water features
Vertical_Distance_To_Hydrology	quantitative	meters	Vert Dist to nearest surface water features
Horizontal_Distance_To_Roadways	quantitative	meters	Horz Dist to nearest roadway
Hillshade_9am	quantitative	0 to 255 index	Hillshade index at 9am, summer solstice
Hillshade_Noon	quantitative	0 to 255 index	Hillshade index at noon, summer soltice
Hillshade_3pm	quantitative	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal_Distance_To_Fire_Points	quantitative	meters	Horz Dist to nearest wildfire ignition points
Wilderness_Area (4 binary columns)	qualitative	0 (absence) or 1	(presence) Wilderness area designation
Soil_Type (40 binary columns)	qualitative	0 (absence) or 1	(presence) Soil Type designation
Cover_Type (7 types)	integer	1 to 7	Forest Cover Type designation

Table 1.1: Dataset description

The target attribute is "Cover_Type" which indicates the type of flora of a specific area, values from 1 to 7 indicate the following types of flora:

1. Spruce/Fir
2. Lodgepole Pine
3. Ponderosa Pine
4. Cottonwood/Willow
5. Aspen
6. Douglas-fir
7. Krummholz

1.3.1 Dataset problems

The dataset has various problems that have been analyzed in an attempt to improve the performance of the predictors.

The *Cover_Type* variable, which is the target value, is distributed as follows:

- Lodgepole Pine: 48.76%
- Spruce/Fir: 36.46%
- Ponderosa Pine: 6.15%
- Krummholz: 3.53%
- Douglas-fir: 2.99%
- Aspen: 1.63%
- Cottonwood/Willow: 0.47%

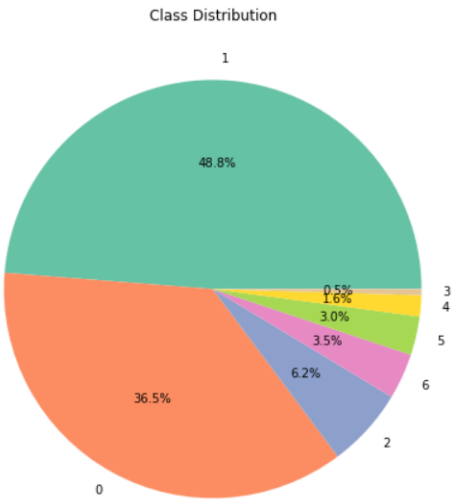

Figure 1.1: Class distribution

This produces an imbalance in the dataset and distorts the results of the predictors.

Another problem is highlighted by the Pearson Correlation Coefficient[3] which indicates how much a variable influences the result of the prediction:

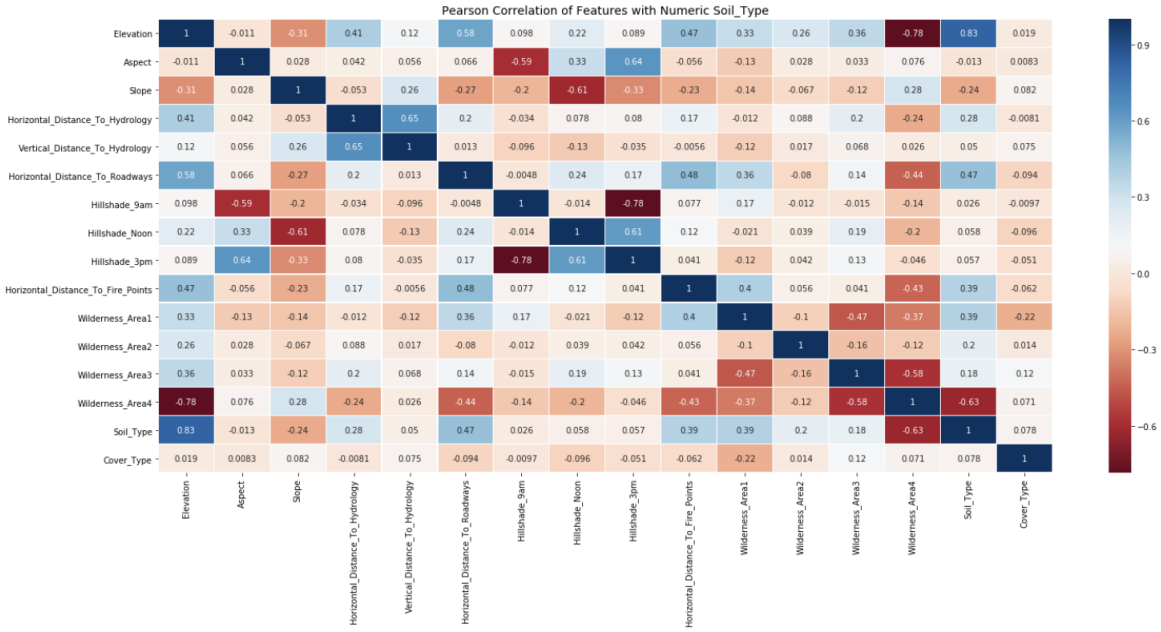

Figure 1.2: Pearson Correlation Coefficient

The dark boxes indicate a greater influence of the data, as you can see a particularly dark area is given by the wilderness_area which greatly affect the result and by the soil_Type. The wilderness_area are distributed as follows:

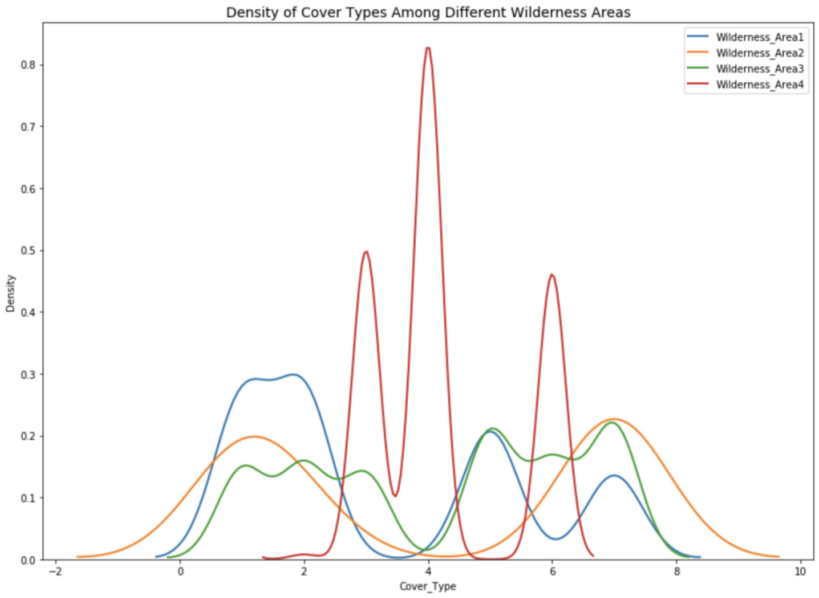

Figure 1.3: Wilderness_area distribution

There is also a strong correlation between the Cover_Type and the combination of the Wilderness_area and the Soil_Type:

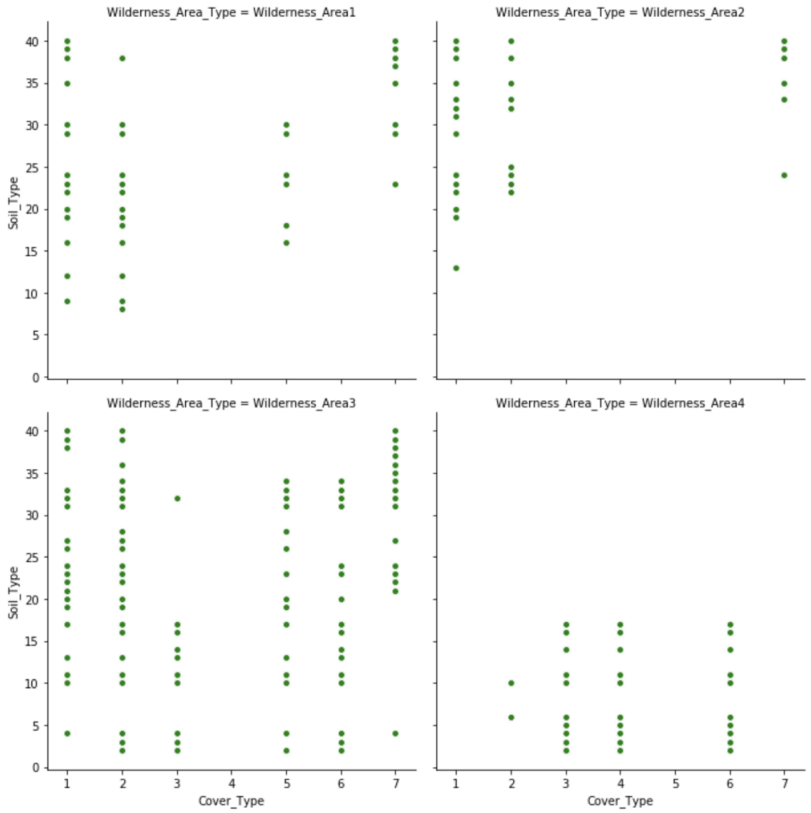

Figure 1.4: Relation between Wilderness_area and Soil_Type

Chapter 2

Predictors

2.1 Online

2.1.1 Ozabag Classifier

Ozabag is an improved version of the Ozabag classifier developed by Nikunj C Oza, a member of NASA's Intelligent Systems Division. This algorithm is part of the Online Bagging algorithms, a version of the bagging algorithms used for data streams that requires only one visit to the training set.

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

Oza developed online versions of bagging and boosting for Data Streams[4]. They show how the process of sampling bootstrap replicates from training data can be simulated in a data stream context. They observe that the probability that any individual example will be chosen for a replicate tends to a Poisson(1) distribution.[5]

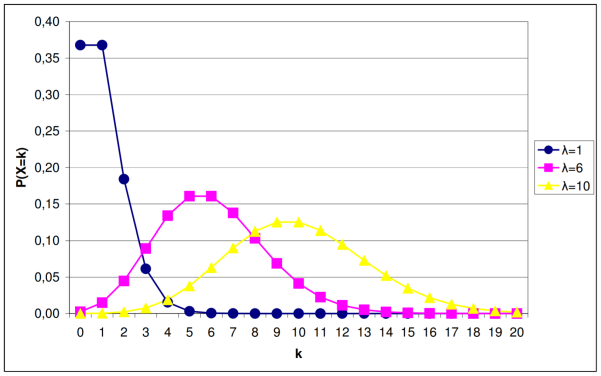

Figure 2.1: Poisson distribution

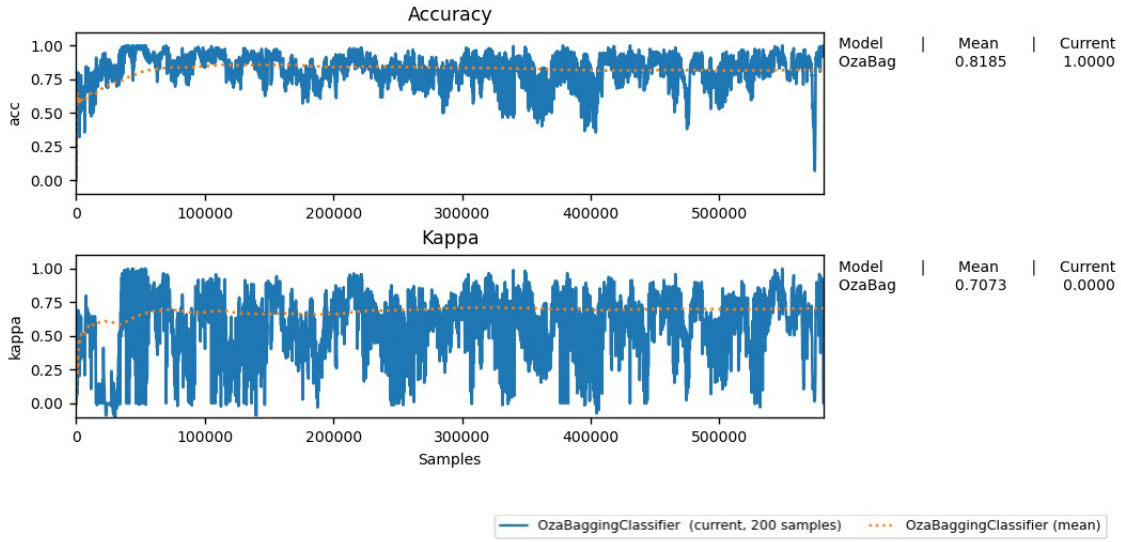

Figure 2.2: OzaBag Classifier results

2.1.2 SAM KNN Classifier

Self Adjusting Memory (SAM) in combination with the k Nearest Neighbor (kNN) classifier is able to cope with heterogeneous concept drift and can be easily applied in practice without any parametrization. It exhibits several analogies to the structure of the human memory as we explicitly partition knowledge among a short- and long-term memory.

In the research field of human memory the dual-store model, consisting of the Short-Term and Long-Term memory (STM & LTM), is largely accepted. Sensory information arrives at the STM and is joined by context relevant knowledge from the LTM. Information getting enough attention by processes such as active rehearsal is transfered into the LTM in form of Synaptic Consolidation. The capacityof the STM is quite limited and information is kept up to one minute, whereas the LTM is able to preserve it for years. [6]

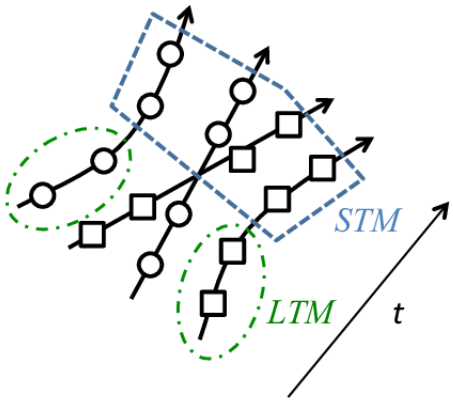

Figure 2.3: SAM model

The SAM architecture is partly inspired by this model and exhibits the following analogies:

- Explicit separation of current and past knowledge, stored in dedicated memories.
- Different conservation spans among the memories.

- Transfer of filtered knowledge from the STM to the LTM.
- Situation dependent usage

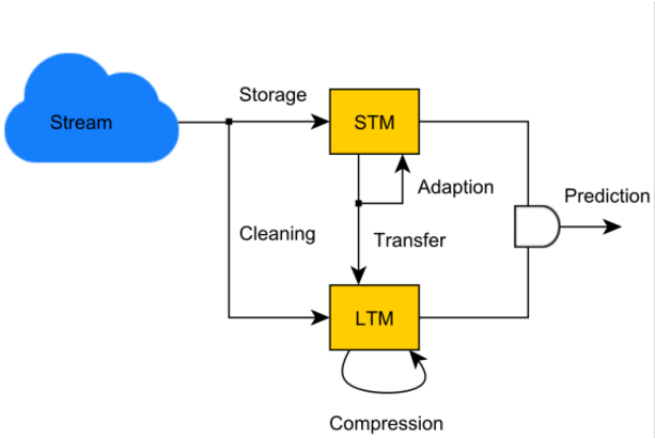

Figure 2.4: SAM architecture

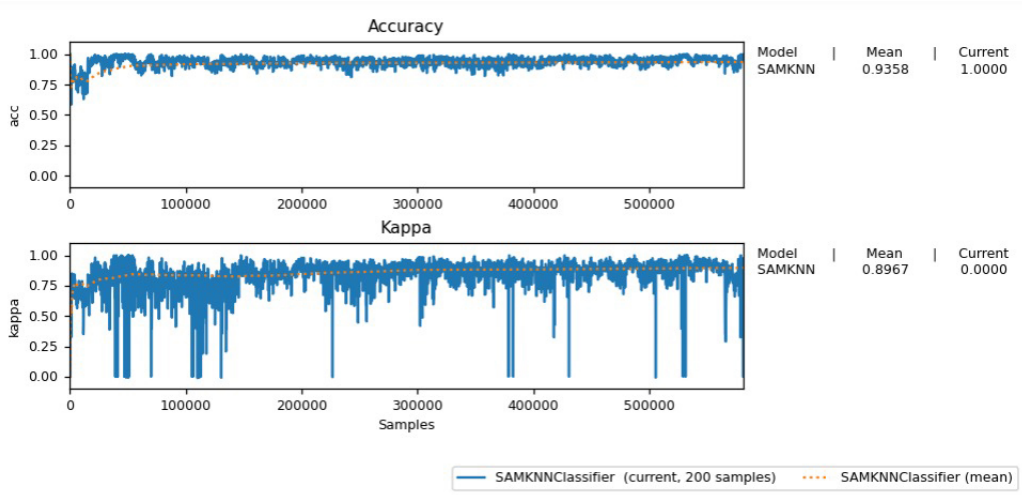

Figure 2.5: SAMKNN Classifier results

2.1.3 Streaming Random Patches Classifier

Streaming Random Patches (SRP) can be viewed as an adaptation of batch learning ensemble methods that combined random samples of instances and random subspaces of features. In the rest of this paper, we refer to random subsets of both features and instances as random patches. Fig 2.4 presents an example of subsampling both instances and features, simultaneously, from streaming data, where only the shaded intersections of the matrix belong to the subsample, i.e., $v1, 1, v2, 1, v6, 1, v1, 3, v2, 3, v6, 3$.

Motivation for exploiting an ensemble of base models trained on random patches is based mainly on the high predictive performance of ensembles for data stream learning that added randomization to the base models by either training them on random samples of instances , random subsets of features or both. We investigate whether selecting the subset of features globally once and before constructing each base model, overcomes locally selecting subsets of features at each node while constructing base trees as in Random Forest.

Bagging is well-known in the machine learning literature for its effect on reducing variance, both in regression and classification, which allows it to perform competitively in a wide range of scenarios, including data streams. In theory, the reduction of the error is strictly related to

$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$v_{1,4}$	$v_{1,5}$	$v_{1,6}$
$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$v_{2,4}$	$v_{2,5}$	$v_{2,6}$
$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$v_{3,4}$	$v_{3,5}$	$v_{3,6}$
$v_{5,1}$	$v_{5,2}$	$v_{5,3}$	$v_{5,4}$	$v_{5,5}$	$v_{5,6}$
$v_{6,1}$	$v_{6,2}$	$v_{6,3}$	$v_{6,4}$	$v_{6,5}$	$v_{6,6}$
$v_{...,1}$	$v_{...,2}$	$v_{...,3}$	$v_{...,4}$	$v_{...,5}$	$v_{...,6}$

Figure 2.6: Representation of a data stream as an unbounded table where the rows are infinite, but the columns are constrained by m input features.

how uncorrelated prediction errors are. Entirely uncorrelated predictions are rarely achievable in practice, yet it is achieved to some extent by encouraging diversity among the learning models.

This itself implies a need to use unstable learners. The standard (batch, unpruned) decision tree is a prime example of an unstable learner: small changes to a training sample can result in remarkably different models, and thus diversity among predictions. Indeed, one readily observes that decision trees are used throughout the literature. In the context of data streams, Hoeffding trees are the popular choice of decision tree, since they are incremental.

However, crucially, Hoeffding trees[7] – unlike their batch counterparts – are in fact stable learners. As far as we are aware we are among the first to focus on this fact in the context of ensembles. Splitting is supported statistically under the Hoeffding bound. This guarantees to a certain (user-specified) confidence level that under a sufficiently large number of examples a Hoeffding tree built incrementally will be equivalent to a batch-built tree. Until such a number of examples is seen, however, Hoeffding trees will not grow and this implies stability. [8]

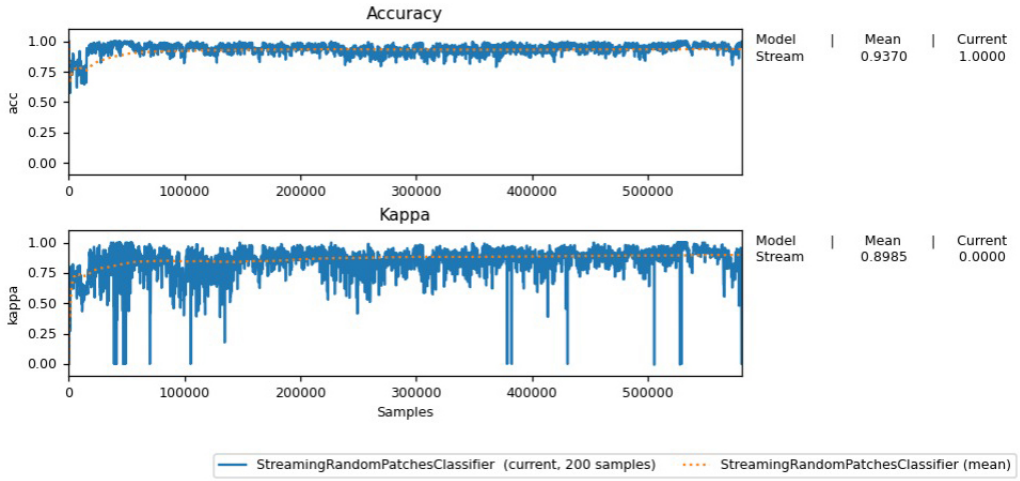

Figure 2.7: Streaming Random Patches Classifier results

2.2 Batch

2.2.1 Preprocessing

As we can see in Section 1.3.1, the dataset is very unbalanced, during the online phase it is not possible to apply balancing techniques as the data arrives in the form of data streams, in the batch phase instead the data are stored statically and it is therefore possible to apply on the balancing technical datasets such as undersampling and oversampling.

Given the imbalance of the dataset, the good results of the predictors are distorted and unbalanced on Cover_Type 1 and 2 as can be seen from the confusion matrix represented below:

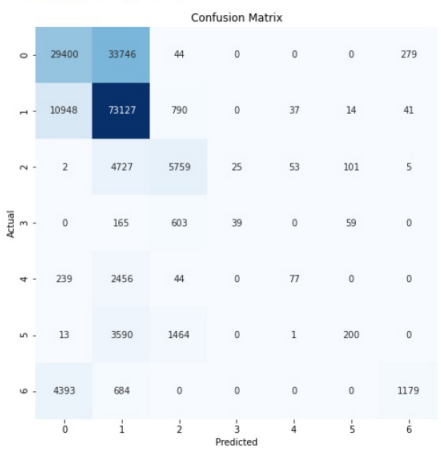

Figure 2.8: SAMNKK Classifier Confusion Matrix

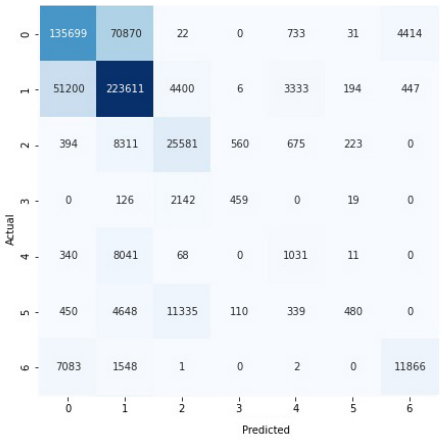

Figure 2.9: OzaBag Classifier Confusion Matrix

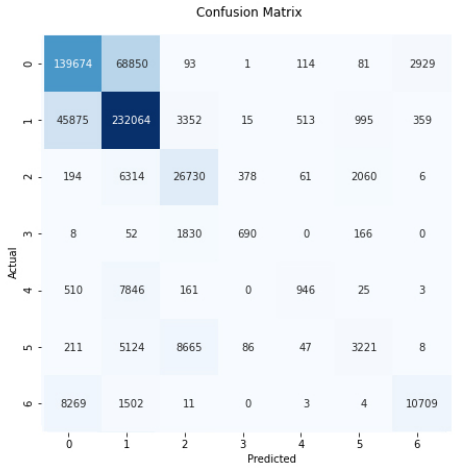

Figure 2.10: Streaming Random Patches Classifier Confusion Matrix

To solve this problem we decided to perform a batch phase in which we first perform an undersapling and then an oversampling and then re-run the predictors.

Scaling

Before performing the Undersampling and Oversampling we performed a Standard scaled of the data which led us to improve performance by 2-5%.

```
def scale(df):
    df = df.copy()
    #Split df into x and y
    y = df['Cover_Type'].copy()
    X = df.drop('Cover_Type', axis = 1).copy()
    scaler = StandardScaler()
    scaler.fit(X)
    X = pd.DataFrame(scaler.transform(X), columns = X.columns)
    return X, y
```

Listing 2.1: Scaling

Undersampling

Then performing the Undersapling on scaled data as follows we get very balanced confusion matrixes:

```
min_class_size = np.min(undersampled_data['Cover_Type'].value_counts().values)
class_subsets = [undersampled_data.query("Cover_Type == "+str(i)) for i in range
    (7)]
for i in range(7):
    class_subsets[i] = class_subsets[i].sample(min_class_size, random_state = 123,
    replace = False)
undersampled_data = pd.concat(class_subsets, axis=0).sample(frac = 1.0,
    random_state = 123).reset_index(drop=True)
```

Listing 2.2: Undersampling

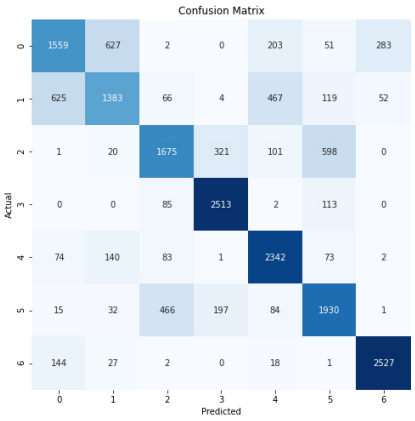

Figure 2.11: SAMNKK Classifier with Undersampling

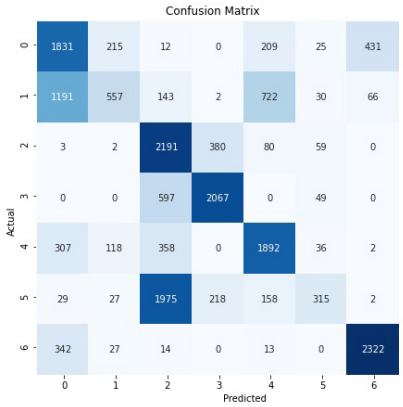

Figure 2.12: OzaBag Classifier with Undersampling

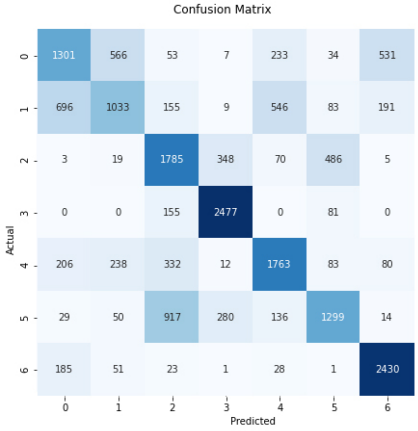

Figure 2.13: Streaming Random Patches Classifier with Undersampling

Oversampling

Then we perform the Oversampling on scaled data to be able to adequately train the predictors with a large number of elements, with the following results:

```
max_class_size = np.max(oversampled_data['Cover_Type'].value_counts().values)
class_subsets = [oversampled_data.query("Cover_Type == "+str(i)) for i in range (7)
    ]
for i in range(7):
    class_subsets[i] = class_subsets[i].sample(max_class_size, random_state = 123,
    replace = True)
```

```
5 oversampled_data = pd.concat(class_subsets, axis=0).sample(frac = 1.0, random_state
      = 123).reset_index(drop=True)
```

Listing 2.3: Oversampling

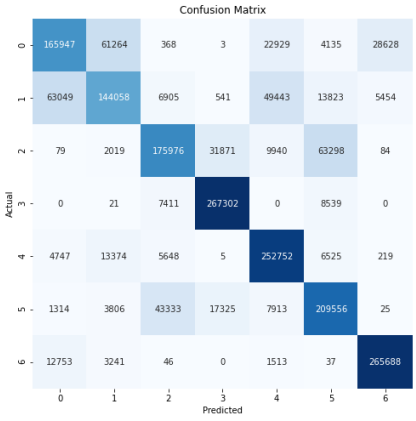

Figure 2.14: SAMNKK Classifier with Oversampling

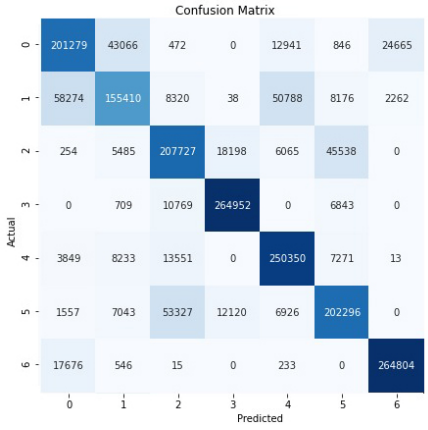

Figure 2.15: OzaBag Classifier with Oversampling

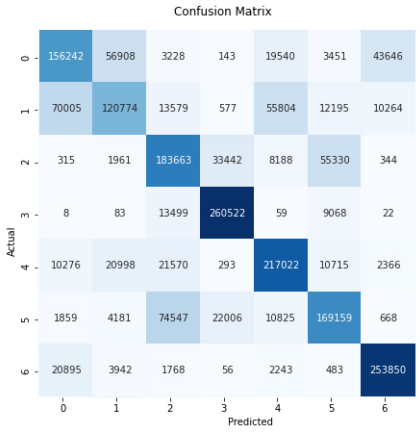

Figure 2.16: Streaming Random Patches Classifier with Oversampling

Execution of classifiers

Finally we re-run classifiers using a for loop that simulates a data stream as shown in the scikit-multiflow [9] documentation:

```
1 while n_samples < max_samples and stream.has_more_samples():
2     X, y = stream.next_sample()
3     y_pred = srp.predict(X)
4     if y[0] == y_pred[0]:
5         correct_cnt += 1
6     srp.partial_fit(X, y)
7     n_samples += 1
```

Listing 2.4: Stream simulation

Chapter 3

Conclusion

The forest covertype dataset is an extremely rich dataset for building multi-class classification model to predict seven different cover types in four different wilderness areas of the Roosevelt National Forest of Northern Colorado. In particular we stated considering a scenario in which a cyberphysical system continuously sends data, collected from the soil to a remote source in order for it to be processed. And so we built three predictors that exploits these three classifiers "**Streaming Random Patches Classifier**", "**SAM KNN Classifier**", "**Ozabag**" to classify the stream of data that arrives from this cyberphisical system in near real time. Even if the result of predictors are very good, we have the problem of unbalanced dataset, indeed as we can see in section 1.3.1, we noticed that most of the samples in the dataset refers to classes "**Lodgepole Pine**", "**Spruce/Fir**", the other five classes have very few samples, and so the learners doesn't learn properly how to predict the other five classes. To resolve this problem we computed some preprocessing in which we applied oversampling and undersampling on dataset to balance it. And so we recomputed the entire process of train and prediction on the stream of data, but now preprocessed, and so even if we noticed that we obtained a decrease in terms of accuracy in all the models in terms of prediction, the models trained with undersampled or oversampled dataset have to be preferred, because predicts all the type of classes in balanced way, and not only two classes as before the preprocessing.

This table provides the final comparison between all the different classifiers before and after the preprocessing:

Predictors	Accuracy	Kappa	Precision	Recall	F1-score
OzaBag	0.81	0,70	0.63	0.65	0.65
SamKnn	0.93	0.89	0.72	0.72	0.72
S.R.P.	0.93	0.89	0.75	0.76	0.75

Table 3.1: Online Phase

Predictors	Accuracy	Kappa	Precision	Recall	F1-score
OzaBag	0.59	0,56	0.54	0.51	0.48
SamKnn	0.73	0.71	0.72	0.72	0.73
S.R.P.	0.64	0.61	0.55	0.56	0.55

Table 3.2: With Undersampling

Predictors	Accuracy	Kappa	Precision	Recall	F1-score
OzaBag	0.78	0,73	0.69	0.68	0.68
SamKnn	0.75	0.73	0.74	0.75	0.74
S.R.P.	0.69	0.66	0.68	0.69	0.68

Table 3.3: With Oversampling

Bibliography

[1] Steven Lennartz and Russell Congalton. “Classifying and mapping forest cover types using IKONOS imagery in the NorthEastern United States”. In: (Oct. 2011).

[2] Jock Blackard and Denis Dean. “Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables”. In: *Computers and Electronics in Agriculture* 24 (Dec. 1999), pp. 131–151. DOI: 10.1016/S0168-1699(99)00046-0.

[3] Philip Sedgwick. “Pearson’s correlation coefficient”. In: *BMJ* 345 (July 2012), e4483–e4483. DOI: 10.1136/bmj.e4483.

[4] Nikunj Oza and Stuart Russell. “Online Bagging and Boosting”. In: *Proceedings of Artificial Intelligence and Statistics* (Jan. 2001).

[5] M. Lafleur et al. “The Poisson Distribution”. In: *The Physics Teacher* 10 (Sept. 1972), pp. 314–321. DOI: 10.1119/1.2352241.

[6] Viktor Losing, Barbara Hammer, and Heiko Wersing. “KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift”. In: Dec. 2016. DOI: 10.1109/ICDM.2016.0040.

[7] Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby. “New options for hoeffding trees”. In: *Australasian Joint Conference on Artificial Intelligence*. Springer. 2007, pp. 90–99.

[8] Heitor Murilo Gomes, Jesse Read, and Albert Bifet. “Streaming Random Patches for Evolving Data Stream Classification”. In: Nov. 2019, pp. 240–249. DOI: 10.1109/ICDM.2019.00034.

[9] Jacob Montiel et al. “Scikit-Multiflow: A Multi-output Streaming Framework”. In: *Journal of Machine Learning Research* 19.72 (2018), pp. 1–5. URL: http://jmlr.org/papers/v19/18-251.html.