Knowledge distillation from random forests

Juan José Contreras González
School of Computer Science and
Electronic Engineering
University of Essex
Colchester, United Kingdom
jc18548@essex.ac.uk

***Abstract* — This report documents the development of a system that will compress a Random Forest Classifier model into a single Decision Tree Classifier through a knowledge distillation process. The experiments conducted will be binary classification problems, and the evaluation of the compression will be by comparing the performance of the decision tree trained through the distillation process with a tree trained with the original dataset.**

The distillation process consists on training a Random Forest Classifier with the dataset and then transferring the knowledge to a single decision tree by the generation of a new dataset for multi-class classification that includes the probabilities, as calculated by the Random Forest Classifier. The new dataset fits the probabilities accordingly through a binning process, and when the model predicts, the bins are divided again to form the original classes from the Random Forest Classifier.

The outcome of this work will contribute to the assessment of model compression in Random Forest Classifiers for binary classification problems

Keywords—Random Forests, Decision Trees, Knowledge Distillation, Model Compression, Data Science

I. INTRODUCTION

A. Handling data

Data has become the central part of most of human activities. Years ago, only big companies handled data, but now, with the existence of personal computers and electronic devices, as well as the spread of wireless communication, almost every individual handles data. Every time a product is bought, a web page is visited, or something is post on social media, data is being generated and shared.

Having this in mind, analysing data and extracting important information from it has become a major discipline, because strategies can be designed in order to make the most out of the knowledge acquired from it. Client behaviour, disease diagnosis, and natural disaster prevention are just examples of applications for this knowledge. To perform this task, patterns within the data must be found.

These patterns can be found with a computer, and an algorithm is required, which is a set of instructions that must be carried out to transform an input into an output. However, there are tasks in which an algorithm cannot be applied, because both input and output are known, but the procedure to transform one into the other is unknown. Determining spam emails from legitimate messages is a good example of this, as the definition of spam messages varies from individual to individual.

When this type of information is unknown, data becomes very important, as what is lacking in knowledge can be made up for in data. Thousands of examples can be compiled, some of which are known to be considered true and some others false to the condition that is considered (in the case of spam email, this means spam or not spam), and the goal is to learn what constitutes a certain outcome.

The process might not be identified completely, but a good and useful approximation can be constructed, and certain patterns or regularities can be detected. This is the main core of Machine Learning.

Machine Learning is programming computers to optimise performance criterion using example data or past experience. The model can be *predictive*, which means it can foresee the future, or *descriptive*, to gain knowledge from data, or a combination of both.

Since the core task of Machine Learning is making an inference from a sample, mathematical models are built through statistics theory, and computers play a two-stage role: first, when training, an algorithm is needed to solve the optimisation problem, and to store and process the existent big amounts of data. Second, once a model is learned, an efficient representation and algorithmic solution for the inference is needed.

When Machine Learning methods are applied to large databases, it is called Data Mining, and its applications are abundant, as it can be applied in finance to use in fraud detection or predictions for the stock market. However, Machine Learning is not only based on databases problems, it is also an important part of artificial intelligence, as for a system to be intelligent, it must have the ability of learning when interacting with a changing environment.

There are different types of learning:

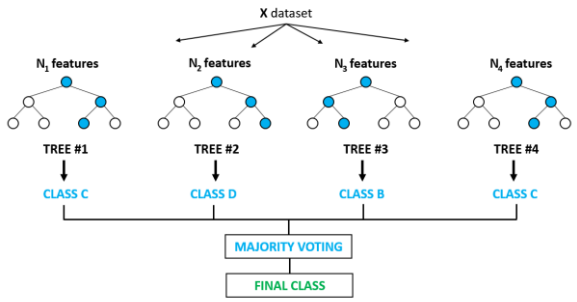
- ***Supervised Learning***: it is performed when there is an input and an output, and the task is to learn the mapping from input to output, but the correct values for this output are known and provided by a supervisor.
- ***Unsupervised Learning***: similar to Supervised Learning, but there is no supervisor and there is only input data, and the aim is to find regularities in this input
- ***Reinforced Learning***: the output of the system is a sequence of actions, and a single one is not important; the important element is the policy that allows the goal to be reached. The program should asses the goodness of policies and learn from past action sequences to be able to generate a policy. [2]

Among Supervised Learning, there are two prediction tasks that can be performed, *Classification* and *Regression*. In classification, the objective is to learn the boundaries separating the instances of one class from the instances of all other classes. This can lead to a separation in just two classes, and is known as binary classification, or in many classes.

Regression, unlike classification, does not provide a Boolean output (the instance belongs to a class or not), but a numeric value. The learning objective in this case is not a class but a numeric function. [8]

B. *Random Forests*

Random Forests are a type of learning algorithm that is an ensemble of Decision Trees, under the premise that combining multiple learning models increases the overall result. Each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest; each one of them casts a unit vote for the most popular class at a certain input. [7]



Using Random Forests is a simple way of taking advantage of parallel computation. However, as it is an ensemble, it could take a big amount of time during the testing stage. Another throwback is the fact that if the individual models are large, as well as the dataset, the computational budget during the training stage becomes very large.

C. *Distillation*

In order to reduce the time required during testing, a special strategy known as "distillation" [1] can be applied. This methodology consists in using a different kind of training once a cumbersome model (in this case the Random Forest) is trained, the knowledge can be transferred to a small model that is more suitable for the deployment.

When a cumbersome model is trained to differ into many classes, the main objective is to maximise the average log probability of the correct answer, but in doing so, the model assigns probabilities to all incorrect answers. These probabilities reflect the generalisation process of the model, and while it is usually accepted that the objective function used for training reflects the true objective, models should not be trained to optimise performance on the training data, but to generalise well to new data. However, training a model to generalise well requires information, and it is not always available.

If a cumbersome model is trained and generalises well, which can happen because it produces an average of a large ensemble of different models, a small model that is trained to generalise in the same way, will perform better on test data than a small model that is trained in a normal way on the same training set that was used to train the ensemble.

An important advantage of transferring knowledge to a smaller model is that a small model can usually be trained on much less data, and with a much higher learning rate than the original cumbersome model. The approach presented for transferring the generalisation ability form the cumbersome model to the smaller model is by using the probabilities obtained from the cumbersome model as soft targets for the other.

II. Methodology

The objective of this work is to implement and evaluate the performance of a Distilled Random Forest Classifier in binary classification problems, in a similar way to the experimented conducted by [6], although they were conducted with Neural Networks.

The process will make use of a Random Forest classifier, and transfer the knowledge acquired to a single tree. This will be done through the Python library scikit-learn and with three different datasets. The knowledge will be transferred into the smaller model with a distillation process by using the probabilities obtained from the cumbersome model as soft targets.

Once the small models are trained by distillation, their performance will be compared with the performance of similar models trained in a traditional fashion in order to evaluate the convenience of their implementation.

A. *Datasets*

The three chosen datasets are the following:

- ***Geographical Origin of Music*** – Contains audio features extracted from 1059 files, created by Fang Zhou at the University of Nottingham, Ningbo, China. [3] The objective is to predict the geographical origin of the music contained. The music is traditional, ethnic or under the 'world' label, as classified by their publishers. No Western music is included, because of its important influence in music from all over the world, and would interfere with the search for aspects of music that most influence location, and the central objective is to specify a location with strong influence on the music.

  The country of origin was determined by the artists' main country or area of residence. The position of the country's capital city was taken as the absolute point of origin, by latitude and longitude. The audio features from the files were extracted with the program MARSYAS, which yields a set of 68 attributes inside of a single vector. Information about chromatic pitches is also included in a different file, with a set of 116 attributes per track. All features were transformed to have a mean of 0 and a standard deviation of 1.

  The dataset is contained in two files, where each file corresponds to a set of features. Both files contain the features for 1059 examples. One of them contains the audio features in the first 68 columns, and the last two correspond to the latitude and longitude of the place of origin of the music, while the other has 116 columns for features and two for the coordinates of the origin of the music.

  The histogram for the distribution of this dataset is presented next:

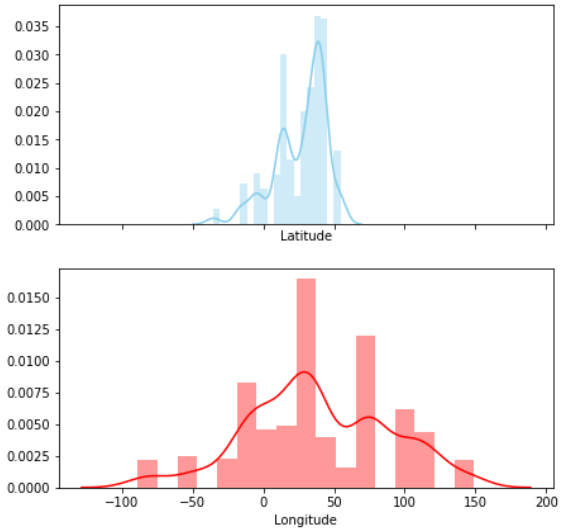


Figure 1. Data distribution for Geographical Origin of Music dataset

The statistic information for the latitude is the following:

Mean: 26.65

Median: 33.66

Var: 340.428

Std: 50.396

For the longitude:

Mean: 38.405

Median: 32.83

Var: 2539.754

Std: 50.396

- ***Year Prediction MSD*** – A prediction of the release year of a song from audio features. Most of the songs are Western, commercial tracks from years between 1922 – 2011, with a peak in the year 2000. It is a subset of the Million Song Dataset, a collaboration between LabROSA and The Echo Nest, prepared by T. Bertin-Mahieux. [4]

  The dataset consists of 515345 examples and 90 attributes. The training and test split must allocate the first 463,715 examples for training and the last 51,630 for testing, as this division ensures that no song from a given artist ends up in both sets.

  From the 90 attributes, 12 correspond to timbre average and the remaining 78 to timbre covariance. The first value in the dataset is the release year of the songs (target), ranging from 1922 to 2011.

  The distribution of the data, corresponding to the year of the songs can be seen in the following histogram.

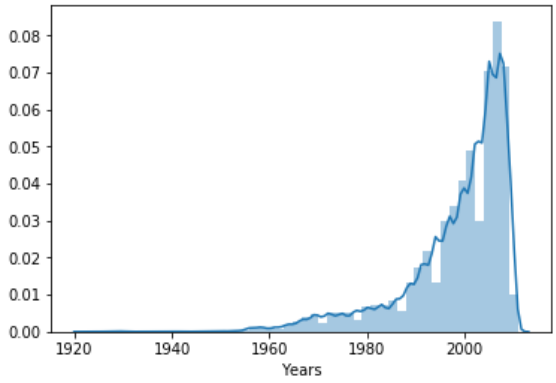


Figure 2. Data distribution for the Year Prediction MSD dataset

The statistic information for this dataset is:

Mean: 1998.397

Median: 2002

Var: 119.487

Std: 10.931

- ***Default of credit card clients*** – The dataset was created by I-Cheng Yeh as part of a research aimed at the case of customers' default payments in Taiwan. [5] The research employed a default payment as a binary response variable, and consists of 23 attributes, containing information regarding amount of given credit, gender, education, marital status, age, history of past payment, amount of bill statement, and amount of previous statement. The labels are given as 1 = Yes and 0 = No for the default payment.

  An analysis of the relationship between the set of attributes via scatter plots, as well as histograms for the distribution of each single variable can be seen in the following chart.

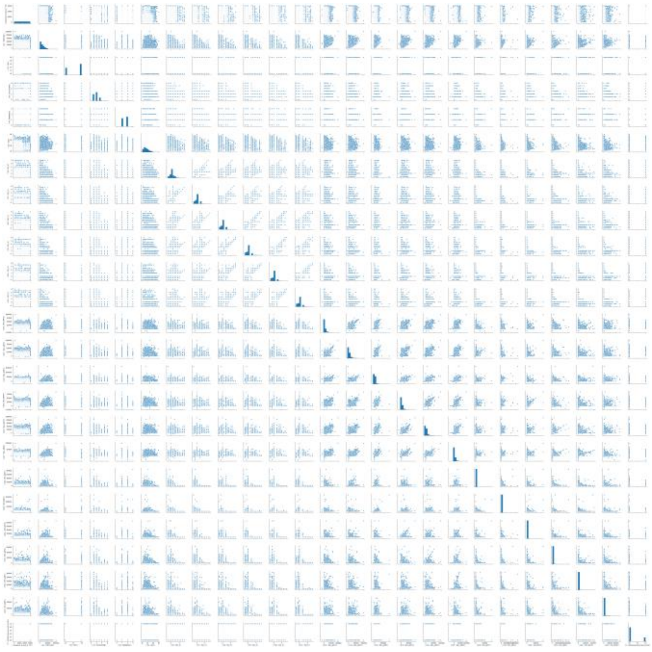


Figure 3. Data distribution and relationships for the Default of credit card clients dataset

The datasets must be revised, to avoid having any anomalies in the data, and performing any pre-processing needed, such as normalisation or deleting empty rows.

B. Methodology

One Random Forest classifier will be trained for each dataset. Each of the three will be constituted by a minimum of 100 trees.

Once the training is complete and the probabilities for each class have been obtained, a new dataset will be created, including the probabilities calculated by the forest, so that it is a multi-class classification dataset with new classes, as shown in the figure:

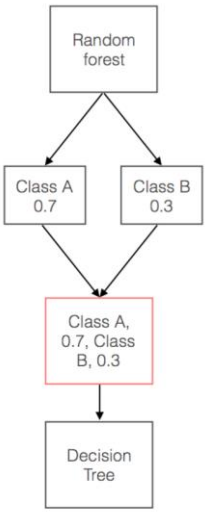


Figure 4. Knowledge transfer

Probability bins will be created as binning method for the data. This will allow the outputs coming from the Random Forest Classifier to sorted in different classes, so that the Decision Tree Classifier learns to classify in more than two classes.

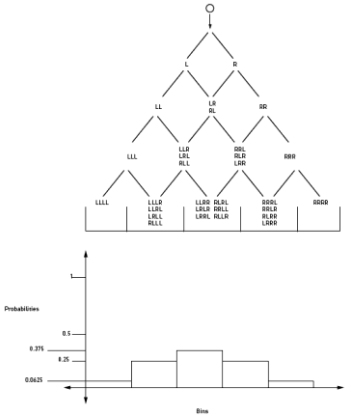


Figure 6. Probability bins

Once the tree learns and predicts examples into these bins, the information will be divided again in two classes. This will be performed considering that every value from the bins that is less or equal to 0.5 will belong to one class, and above 0.5 to the other class.

Decision tree classifiers will be trained and tested with this new dataset, in order to asses their accuracy on the original datasets. The outcome will be compared to the original model in terms of size, shape and performance.

C. Hyperparameters tuning

The random forest will be generated, trained and tested through the scikit learn Python library. This library allows the splitting of data into training and validation, as well as performing cross validation multiple times in order to obtain the best hyperparameters for the model.

To tune the hyperparameters of the Random Forest Classifier, a Cross Validation Grid Search will be conducted, which will perform an exhaustive search over specified parameter values for an estimator. Fitting this grid search will return a set of best parameters for the model.

This process must be performed for each of the three datasets, as the same hyperparameters will most likely not give the best result for two different cases.

Both, the random forest classifier and the decision tree classifiers will go through this hyperparameter tuning process.

III. Experiments

Three experiments will be performed in this work, one for each of the abovementioned datasets. The main goal for each of the experiments is to be able to correctly classify examples into two classes through the distillation approach for a Random Forest Classifier, so the datasets must be adjusted so that all the information can be fixed in two classes.

A. Previous studies and results

The process of transferring information from a cumbersome model to a smaller model has been performed by [1] in speech recognition, through Deep Neural Networks. In their experiments, a model used by Android voice search was set as baseline, 10 separate models of prediction and a distilled single model were compared.

The 10 separate models used the exact same architecture as the baseline and were initialised with random initial parameter values. The averaged predictions of the 10 models were used to create the soft targets.

The results obtained by these experiments were based on Test Frame Accuracy and Word Error Rate. They are shown in the following table.

System	Test Frame Accuracy	WER
Baseline	58.9%	10.9%
10xEnsemble	61.1%	10.7%
Distilled Single Model	60.8%	10.7%

Table 1. DNN Results

It can be seen that the distilled single model and the 10-model ensemble give very similar results and they both outperform the baseline model. This information is crucial for the development of the work presented in this paper, as a Random Forest is a model that consists of the results of

various single models, so the improvement over a baseline simple model will be present during training, but this already improved knowledge will then be subjected to a distillation process, so that the results can be further improved.

[5] conducted a set of experiments of compressed modelling, they compared the performance of the compressed models with the performance of the target ensemble selection models on eight test problems. In their experiments, they compressed the ensemble selection models and MUNGE data. The results reflected that the performance of the mimic models was as good or better than the ensemble models, but always better than the ones trained on the original data.

An important outcome from their research is the fact that with compression, the size of the training set can be made arbitrarily large, to avoid overfitting, and reliably measure the effect of the model's size on performance.

B. Setup

The dataset corresponding to the default of credit card clients is already in a form for binary classification, as the labels given are 'Yes' and 'No' for the default payment, so no pre-processing is needed.

On the other hand, both Year prediction MSD and Geographical origin of music datasets are in a format that yields regression problems, rather than classification. However, this can be solved by fixing thresholds that divide the output values in two classes.

For the year prediction, a year will be set as threshold so that if the example belongs to an earlier date, it will be assigned as 'class A', otherwise, it will be labelled as 'class B'.

In the case of the geographical origin, besides grouping the values in two classes, there is another characteristic of the dataset that must be adjusted, which is the fact that the labels are pairs of coordinates (latitude and longitude). The procedure will then be considering the absolute distance from a fixed arbitrary location to the given coordinates of the examples.

The problem then becomes a binary classification in order to determine if the origin of the specified music example. If the distance is smaller than a specified threshold, the example corresponds to class A, otherwise, it belongs to class B.

IV. Discussion

The performance of the system will be evaluated through different metrics. The first one is the accuracy of the system, because it reflects the percentage of unseen examples that it can classify correctly. However, this metric is not enough to fully evaluate the system, because the behaviour of the system is not understood completely unless a relationship between predictions and the correct values is present. This can be achieved through the inclusion of recall and precision.

The balance of recall and precision can be expressed as the harmonic mean of both values, and it is commonly referred to as F1 score.

The performance of the classifier will also be assessed by the Kappa Statistic, as it states a proportion between its predictions and the predictions that would be expected from a perfect predictor.

Once these values are obtained, the difference between the results of the decision tree trained with the distilled information from the random forest and the results from the one trained normally will be compared.

The size of the trees will also be considered for comparison, as it is another metric for complexity, as well as the time required for training.

V. Conclusion

Common approaches for Machine Learning and Data Analysis have been used for some time and yielded good results. However, there are opportunities for new methods to be developed, as well as improvements to the most common ones. These improvements can be achieved by modifying certain steps during the implementation of the models, or by generating hybrid systems.

The distillation process is an approach that involves an adaptation process in the training of a simple model, as it is a transfer of the knowledge acquired by a cumbersome model into a smaller model, and its implementation derives promising results as these models perform better on test data than models that is trained in a normal way on the same training set that was used to train the cumbersome model. It is also important to mention that a small model can be trained on less data and at a higher learning rate than a bigger model.

The process has been used by [1] in a Deep Neural Network model, and good results were achieved, as the simpler neural network trained through distillation was easier to deploy than the original.

Therefore, evaluating the performance of the distillation process in Random Forests classifiers will produce important information about optimising a decision process in real-time and fast deployment applications. This is specially taking into consideration the fact that, as stated by [1] in their experiments, a weighted average of 10 models outperforms a simple model, but is almost as good as a distilled model, and since Random forests are outcomes of multiple models, this outperformance will be obtained during the initial knowledge acquisition, and is expected to improve even more when transferred to the simple model.

VI. References

[1] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[2] Cowling, P. I., Powley, E. J., & Whitehouse, D. (2012). Information set monte carlo tree search. IEEE Transactions on Computational Intelligence and AI in Games, 4(2), 120-143.

[3] Fang Zhou, Claire Q and Ross. D. King. Predicting the Geographical Origin of Music, ICDM, 2014

[4] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[5] Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480.

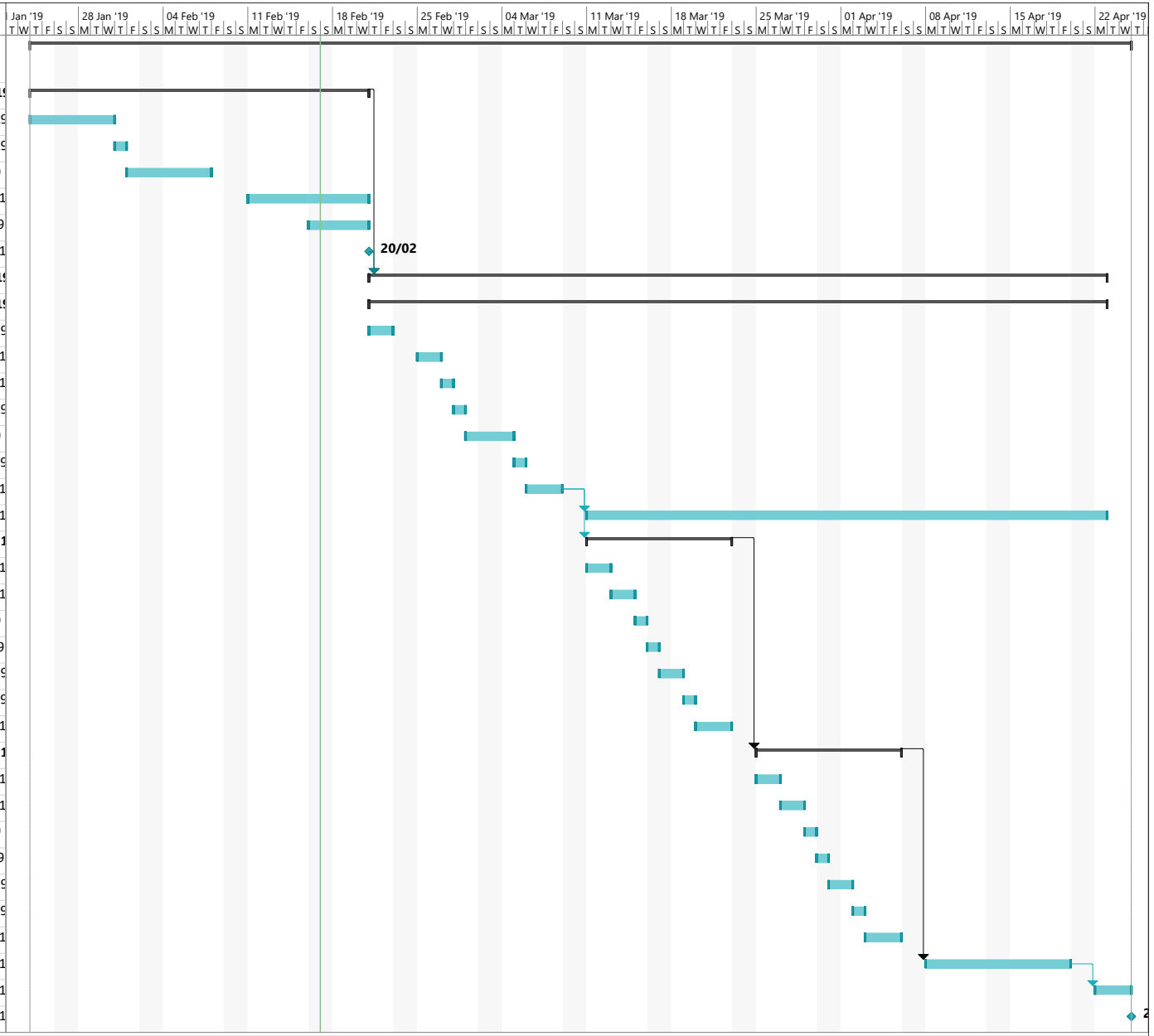
[6] Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression." Proceedings of the 12th ACM SIGKDD

international conference on Knowledge discovery and data mining. ACM, 2006.

[7] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[8] Alpaydin, Ethem. Machine learning: the new AI. MIT press, 2016.

ID	Task Mode	Task Name	Duration	Start
1		**Knowledge Distillation from Random Forests**	**65 days**	**Thu 24/01/19**
2		**First assignment**	**20 days**	**Thu 24/01/19**
3		Task understanding	5 days	Thu 24/01/19
4		Dataset inspection	1 day	Thu 31/01/19
5		Literature Review	5 days	Fri 01/02/19
6		First report	8 days	Mon 11/02/19
7		Code propotyping	4 days	Sat 16/02/19
8		First submission	0 days	Wed 20/02/19
9		**Second assignment**	**43 days**	**Thu 21/02/19**
10		**Dataset 1**	**43 days**	**Thu 21/02/19**
11		Dataset preparation	2 days	Thu 21/02/19
12		Model design	2 days	Mon 25/02/19
13		Random Forest training	1 day	Wed 27/02/19
14		New dataset creation	1 day	Thu 28/02/19
15		Decision trees training	2 days	Fri 01/03/19
16		Test and validation	1 day	Tue 05/03/19
17		Analysis of results	3 days	Wed 06/03/19
18		Final report	31 days	Mon 11/03/19
19		**Dataset 3**	**10 days**	**Mon 11/03/19**
20		Dataset preparation	2 days	Mon 11/03/19
21		Model design	2 days	Wed 13/03/19
22		Random Forest training	1 day	Fri 15/03/19
23		New dataset creation	1 day	Sat 16/03/19
24		Decision trees training	2 days	Sun 17/03/19
25		Test and validation	1 day	Tue 19/03/19
26		Analysis of results	3 days	Wed 20/03/19
27		**Dataset 2**	**10 days**	**Mon 25/03/19**
28		Dataset preparation	2 days	Mon 25/03/19
29		Model design	2 days	Wed 27/03/19
30		Random Forest training	1 day	Fri 29/03/19
31		New dataset creation	1 day	Sat 30/03/19
32		Decision trees training	2 days	Sun 31/03/19
33		Test and validation	1 day	Tue 02/04/19
34		Analysis of results	3 days	Wed 03/04/19
35		Problem solving	10 days	Mon 08/04/19
36		Final review	3 days	Mon 22/04/19
37		Final submission	0 days	Wed 24/04/19



Project: CE888 Gantt Chart
Date: Sun 17/02/19

Task	Project Summary	Manual Task	Start-only	Deadline
Split	Inactive Task	Duration-only	Finish-only	Progress
Milestone	Inactive Milestone	Manual Summary Rollup	External Tasks	Manual Progress
Summary	Inactive Summary	Manual Summary	External Milestone

In [ ]:
```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score as acc
from sklearn.preprocessing import KBinsDiscretizer
```

In [ ]:
```
#prepare data
df = pd.read_csv("./YearPredictionMSD.txt", header = None, delimiter = ",")
X = df.loc[:,1:]
y = df.loc[:,0]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.70, random_state=5) #70% training  30% test
```

In [ ]:
```
#train random forest
clf=RandomForestClassifier(n_estimators=100)
clf.fit(X_train,y_train)

#Predict probabilities
y_prob_pred = clf.predict_proba(X_test)

#Predict in all data
y_prob_pred = clf.predict_proba(X)
```

In [ ]:
```
#Create new dataset

#fit in bins
ds = []
bins = KBinsDiscretizer(n_bins=10, encode='ordinal', strategy='uniform')
yt = bins.fit_transform(y_prob_pred[0])
#yt = bins.transform(y_prob_pred[0])

X_train_new, X_test_new, y_train_new, y_test_new = train_test_split(X_train, yt, test_size=0.70, random_state=5) #70% training  30% test
```

In [ ]:
```
#Generate tree
tree = tree.DecisionTreeClassifier()
tree.fit(X_train_new, y_train_new)

#Predict
y_pred_tree = tree.predict(X_test_new)

#Return bins to two classes
for i in range(len(y_pred_tree)):
    if(y_pred_tree[i] <= 0.5):
        y_pred_tree[i] = 0
    else:
        y_pred_tree[i] = 1

#Test
print("Accuracy:", acc(y_test, y_pred_tree))

#Predict original
y_pred = tree.predict(X_test)

#Test original
print("Accuracy:", acc(y_test, y_pred))
```