Capstone Project - Machine Learning Engineer Nanodegree - Forest Cover Type Prediction (https://www.kaggle.com/c/forest-cover-type-prediction)

Brandon Ma

Feb., 2017

Definition

Project Overview

People depend on forests to live. They filter the water we drink and the air we breathe. Worldwide, 1.6 billion people rely on forests for their livelihoods, including food, clothing, or shelter. Forests are home to nearly half of the world's species, including some of the most endangered birds and mammals, such as orangutans, gorillas, pandas, Northern Spotted Owls and Marbled Murrelets. Deforestation and forest destruction is the second leading cause of carbon pollution, causing 20% of total greenhouse gas emissions.

In the US, much of our forestland is private. If landowners can't earn a living from these forests, they will inevitably cut them down for farms, ranches or real estate development. While total acreage of forests in the US remains relatively stable, certain parts of the country are seeing declining forest coverage. For example, the US Forest Service estimates 12 million acres of forest in the Southeast will be lost to suburban real estate development between 1992 - 2020. Many states in the US, have designed 'Forest classification' programs to to help keep the private forests intact. It allows landowners with certain acres of forest to set it aside and to remain as forest. In return for meeting program guidelines landowners receive property tax breaks, forestry literature and periodic free inspections by a professional forester while the forest is enrolled in the program.

The goal of this project is to create a model that can be used to predict and classify forests. Given the strictly cartographic variables (as opposed to remotely sensed data), this project tries to evaluate the accuracy of forest type prediction. To evaluate, the mean accuracy on the given test data and labels was used. Kaggle has launched a machine learning competition on this topic.

The study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. Each observation is a 30m x 30m patch. You are asked to predict an integer classification for the forest cover type. The seven types are:
1 - Spruce/Fir
2 - Lodgepole Pine
3 - Ponderosa Pine

4 - Cottonwood/Willow
5 - Aspen
6 - Douglas-fir
7 - Krummholz

The training set (15120 observations) contains both features and the Cover_Type. The test set contains only the features. You must predict the Cover_Type for every row in the test set (565892 observations).

Problem Statement

This project is looking to solve the problem of predicting forest cover type. As mentioned, our training set has 15120 data points. Each data point falls into one of seven categories. These data points make up the training dataset used for fitting our machine learning models. The models is then used on an unlabeled test dataset. A predicted score of 1-7 will be assigned to each test data point. The aim is to accurately score the test data and submit to kaggle competition. While the competition has passed its deadline and is inactive, test data predictions can still be submitted for benchmarking on the leaderboard, and ranked the performance of the algorithms.

After exploring the dataset and understanding its structure, it became apparent that this is a classification problem. Numerous attempts were made on this Kaggle competition. I decided to base this project on the knowledge I learned from Udacity. The solution may be far from perfect, but one I can speak to comfortably.

The approach taken in this project is as follow:
- pre-process the data to remove outliers
- fit various models
- tune the models if needed with grid search cross validation techniques
- perform predictions on the test dataset
- ensemble the models predictions to get a final score
- submit the test dataset to the Kaggle competition for benchmarking and evaluation.

Analysis

Data Exploration

The actual forest cover type for a given 30 x 30 meter cell was determined from US Forest Service (USFS) Region 2 Resource Information System data. Independent variables were then derived from data obtained from the US Geological Survey and USFS. The data is in raw form

(not scaled) and contains binary columns of data for qualitative independent variables such as wilderness areas and soil type.

This study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. These areas represent forests with minimal human-caused disturbances, so that existing forest cover types are more a result of ecological processes rather than forest management practices.

The study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. Each observation is a 30m x 30m patch. You are asked to predict an integer classification for the forest cover type. The seven types are:

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

The training set (15120 observations) contains both features and the Cover_Type. The test set contains only the features. You must predict the Cover_Type for every row in the test set (565892 observations).

Data Fields

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

The wilderness areas are:

1 - Rawah Wilderness Area
2 - Neota Wilderness Area
3 - Comanche Peak Wilderness Area
4 - Cache la Poudre Wilderness Area

The soil types are:

1 Cathedral family - Rock outcrop complex, extremely stony.
2 Vanet - Ratake families complex, very stony.
3 Haploborolis - Rock outcrop complex, rubbly.
4 Ratake family - Rock outcrop complex, rubbly.
5 Vanet family - Rock outcrop complex complex, rubbly.
6 Vanet - Wetmore families - Rock outcrop complex, stony.
7 Gothic family.
8 Supervisor - Limber families complex.
9 Troutville family, very stony.
10 Bullwark - Catamount families - Rock outcrop complex, rubbly.
11 Bullwark - Catamount families - Rock land complex, rubbly.
12 Legault family - Rock land complex, stony.
13 Catamount family - Rock land - Bullwark family complex, rubbly.
14 Pachic Argiborolis - Aquolis complex.
15 unspecified in the USFS Soil and ELU Survey.
16 Cryaquolis - Cryoborolis complex.
17 Gateview family - Cryaquolis complex.
18 Rogert family, very stony.
19 Typic Cryaquolis - Borohemists complex.
20 Typic Cryaquepts - Typic Cryaquolls complex.
21 Typic Cryaquolls - Leighcan family, till substratum complex.
22 Leighcan family, till substratum, extremely bouldery.
23 Leighcan family, till substratum - Typic Cryaquolls complex.
24 Leighcan family, extremely stony.
25 Leighcan family, warm, extremely stony.
26 Granile - Catamount families complex, very stony.
27 Leighcan family, warm - Rock outcrop complex, extremely stony.
28 Leighcan family - Rock outcrop complex, extremely stony.
29 Como - Legault families complex, extremely stony.
30 Como family - Rock land - Legault family complex, extremely stony.
31 Leighcan - Catamount families complex, extremely stony.
32 Catamount family - Rock outcrop - Leighcan family complex, extremely stony.
33 Leighcan - Catamount families - Rock outcrop complex, extremely stony.
34 Cryorthents - Rock land complex, extremely stony.
35 Cryumbrepts - Rock outcrop - Cryaquepts complex.
36 Bross family - Rock land - Cryumbrepts complex, extremely stony.

37 Rock outcrop - Cryumbrepts - Cryorthents complex, extremely stony.
38 Leighcan - Moran families - Cryaquolls complex, extremely stony.
39 Moran family - Cryorthents - Leighcan family complex, extremely stony.
40 Moran family - Cryorthents - Rock land complex, extremely stony.

Diving into the dataset some more, we learned that there are 54 features. No attribute is missing as the count in each attribute equals the size of the dataset. Wilderness_Area and Soil_Type are one hot encoded. Soil_Type7 and Soil_Type15 can be removed from the dataset because they are constant.

We also learned that the dataset is not scaled the same. Several features in Soil_Type show a large skew. Hence, rescaling and normalization may be necessary.

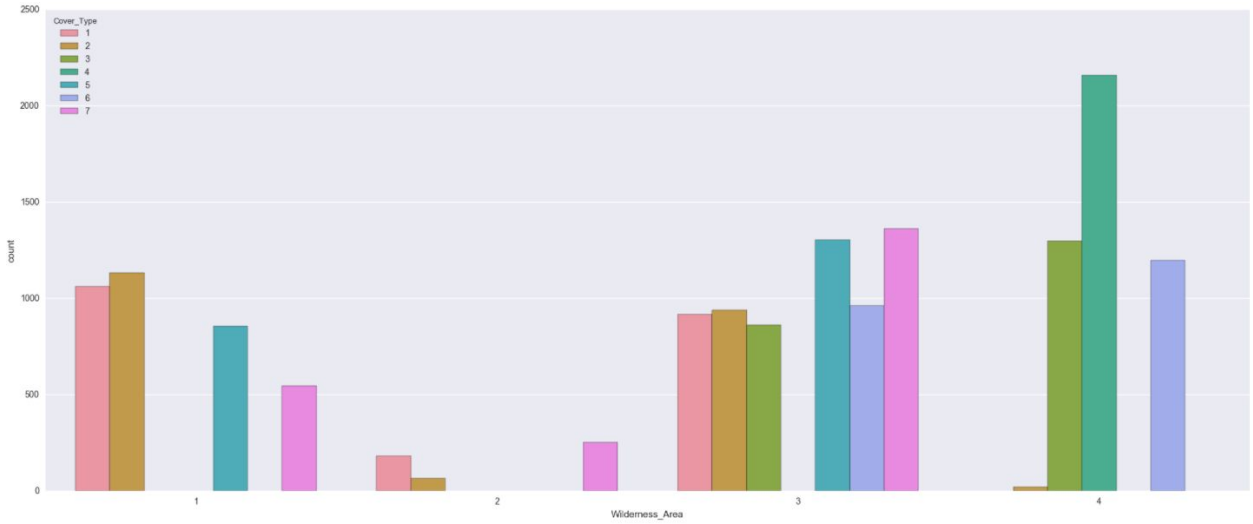
```
Soil_Type34                          26.161230
Soil_Type35                          12.052838
Soil_Type36                          38.849712
Soil_Type37                          21.018939
Soil_Type38                           4.221771
Soil_Type39                           4.479186
Soil_Type40                           5.475256
```

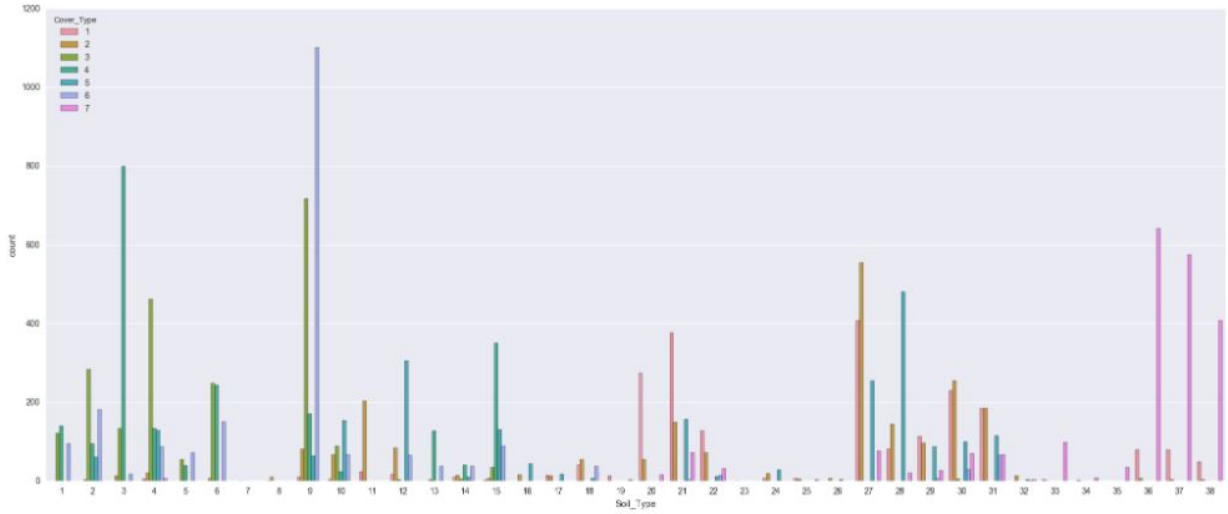
The number of instances belonging to each class is equally presented. Hence, no re-balancing is necessary.

```
dataset.groupby('Cover_Type').size()
Cover_Type
1   2160
2   2160
3   2160
4   2160
5   2160
6   2160
7   2160
```

Exploration Visualization

Here I am converting the one-hot encoded features, wilderness and soil type, back into a single variable. The data was visualized using seaborn.countplot. A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The figures below show something interesting here. WildernessArea_4 has a lot of cover_type 4 in presence, distinguishing from other cover types. WildernessArea3 has not shown much distinction. Several soil type also shows this similar property.





Algorithms and Techniques

I decide to use ensembling techniques for this project. Ensembling is a general term for combining many classifiers by averaging or voting. It is a form of meta learning in that it focuses on how to merge results of arbitrary underlying classifiers. For the models themselves, I am using Bagged Decision Tree classifier, K Nearest Neighbor(KNN) classifier, Stochastic Gradient Descent.

The ensembling technique allows me to combine the above algorithms such that the prediction accuracy is better than each individual one alone. I used a weighted model average scoring method. A random set of weights were used to vary the output of each model. The weights produced the highest score was used on the test dataset for final submission.

Bagged Decision Tree are generally used in remote data sensing, classify different types of land: mountainous terrain, urban terrain, agricultural lands or Wetlands. Strengths and Weaknesses of the algorithm:

Pros:
- reduces variance in comparison to regular decision trees
- Can provide variable importance measures
  - classification: Gini index
  - regression: RSS
- Can easily handle qualitative (categorical) features
- Out of bag (OOB) estimates can be used for model validation

Cons:
- Not as easy to visually interpret
- Does not reduce variance if the features are correlated

This model could deal with classification problem with small data samples. The variance reducing ability might be especially useful in this case. Bagged tree can easily handle qualitative and categorical features. The given data has both.

KNN is commonly used in classifying data into clusters in unsupervised learning; recommend a similar product. Strengths and Weaknesses of the model:

Pros:
- simple, cheap, no training involved("lazy")
- naturally handle multiclass classification and regression

Cons:
- expensive and slow predicting new instances
- distance function must be difined in a meaningful way to data
- perform poorly on high dimension dataset

This model is a natural pick for classification problem. As mentioned in the sklearn "Choosing the right estimator"(http://scikit-learn.org/stable/tutorial/machine_learning_map/), it is the first picked algorithm. I don't want to pick a complex model, while the problem could be solved be a simpler one. As we shall not assume the data given is complex or not in the first place.

The last model is an estimator that implements regularized linear models with stochastic gradient descent (SGD) learning. It is a simple yet very efficient approach to discriminative learning of linear classifiers, similar to SVMs. The fact that it was a fast and efficient algorithm and that it supports multi-class classification by combining multiple binary classifiers in a “one versus all” (OVA) scheme solidified its choosing.

The input training data will preprocessed and vectorized, and then being handled by the SVM, NB and SGD models. Using cross validation techniques on the training data, five splits will be made and used to test the models' fit on the accuracy score. After the models are tuned, they will then be fitted with the whole training dataset and used to predict the test dataset. The output

of the models will be ensembled using a weighted average approach to produce the final submission.

Benchmark

The benchmark for this project is taken from the Kaggle competition leader board. While the winner's score is 1, I decided to use my initial submit score of 0.66 as a benchmark and improve from there.

Methodology

Data Preprocessing

There were some issues with the dataset. Some features in the dataset show zero standard deviation. These features should be dropped from the dataset. Of course, the 'Id' columns will be dropped as well. Since it is not useful for training the model. The numerical values in the dataset are skewed. Normalization on the numerical values in the dataset was carried out during data preprocessing.

Implementation

There are three algorithms used as described above: Bagged DT, KNN and SGDC. They all took in preprocessed data for fitting and prediction. To empirically evaluate each model, I used a ten fold cross validation technique on the training dataset. This splits the training dataset into ten parts, each with its own training and testing with features and labels. Cross validation is crucial to avoid overfitting the data. Each model was assigned an average score that could be used as a reference in the ensembling later.

For Bagged DT, a max_depth of 13 was chosen for the base DT estimator. The bagging classifier, when using 100 estimators gave the best score. KNN was using 1 near neighbor. Bagged DT runs a bit slower then KNN and SGDC. Since the dataset only has around 50 features, all models were relatively easy to work with.

The cross validation outputs were used in the ensembling phase to determine the weights to be assigned to each model. The weight vector were generated using numpy Dirichlet distribution. Each weight vector consist of three numbers summing to 1. The maximum weight were selected by looking for the maximum f1 score. This max weight was then applied respectively to each model's prediction of the test dataset. These predictions along with their corresponding data ID

were submitted to the Kaggle leaderboard for final evaluation. The models were tuned to optimized for high f1 score and Kaggle evaluation score.

The following is the implementation steps in this project:

1. Load datasets in pandas dataframe.
2. Preprocess the dataset,decide which columns to use.
3. Normalize numerical values, concatenate with categorical values.
4. Assign cover type as the target label.
5. Split the training dataset using the Stratified K-fold function for model cross validation.
6. Fit the models with preprocessed data.
7. Predict the cover type target.
8. Compute the f1 score.
9. Tune the model parameters, repeat the above step to get a satisfactory score.
10. Update the model parameters for final predictions of the test dataset.
11. Preprocess the training dataset and fit to the models.
12. Predict the cover type from the test dataset.
13. Finding weights to be used on the three models by optimizing the highest f1 score.
14. Use the max weights to ensemble all three outputs.
15. Submit to Kaggle competition for evaluation.

Refinement

The main part that were refined for this project: the models.

The models were tuned using a Grid Search Cross Validation technique that exhaustively searches over specified parameter values. A dictionary object would be inputted, full of various parameter combination, to the Grid Search function that would test all possible combinations. It would optimize for the highest f1 score.

For Bagged DT, the tuning parameters are:

- decision tree max depth. (max_depth)
- number of estimators. (n_estimators)

For KNN, the tuning parameters are:

- Algorithm. ("auto", "ball_tree", "kd_tree", "brute")
- Number of neighbors. (n_neighbors)

For SGDC, the tuning parameters are:

- Loss function. ("hinge", "log", "modified_huber", "squared_hinge", "perceptron")
- Penalty. ("l2", "l1", "elasticnet")

The score before and after tuning:

Model	Before Tuning	After Tuning
Bagged DT	0.561	0.771
KNN	0.669	0.733
SGDC	0.588	0.610

Results

Model Evaluation and Validation

The aim is to accurately predict the forest cover type by improving the Kaggle competition's benchmark score. To get the final submission, each individual model was tuned using Grid Search Cross Validation to find its best parameters. Then they were evaluated by performing a ten fold Cross Validation fit and predict on the training dataset.

The following are the f1 scores for each of the final models along with their tuned parameters.

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini',
 max_depth=13,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=19, splitter='best'),
         bootstrap=True, bootstrap_features=False, max_features=1.0,
         max_samples=1.0, n_estimators=100, n_jobs=-1, oob_score=False,
         random_state=19, verbose=0, warm_start=False)
Kfold score 1: 0.549603174603
Kfold score 2: 0.637566137566
Kfold score 3: 0.62037037037
Kfold score 4: 0.569444444444
Kfold score 5: 0.580687830688
Kfold score 6: 0.499338624339
Kfold score 7: 0.555555555556
Kfold score 8: 0.683201058201
Kfold score 9: 0.736111111111
Kfold score 10: 0.630291005291
Average score: 0.606216931217
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=-1, n_neighbors=1, p=2,
           weights='uniform')
```

```
/Users/brandon/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/utils/validation.py:42
9: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardSca
ler.
  warnings.warn(msg, _DataConversionWarning)
```

```
Kfold score 1: 0.634259259259
Kfold score 2: 0.67328042328
Kfold score 3: 0.699735449735
Kfold score 4: 0.640211640212
Kfold score 5: 0.622354497354
Kfold score 6: 0.636243386243
Kfold score 7: 0.67791005291
Kfold score 8: 0.664021164021
Kfold score 9: 0.787698412698
Kfold score 10: 0.739417989418
Average score: 0.677513227513
SGDClassifier(alpha=0.0001, average=False, class_weight=None, epsilon=0.1,
       eta0=0.0, fit_intercept=True, l1_ratio=0.15,
       learning_rate='optimal', loss='log', n_iter=7, n_jobs=1,
       penalty='elasticnet', power_t=0.5, random_state=19, shuffle=True,
       verbose=0, warm_start=False)

SGDClassifier(alpha=0.0001, average=False, class_weight=None, epsilon=0.1,
       eta0=0.0, fit_intercept=True, l1_ratio=0.15,
       learning_rate='optimal', loss='log', n_iter=7, n_jobs=1,
       penalty='elasticnet', power_t=0.5, random_state=19, shuffle=True,
       verbose=0, warm_start=False)
```

```
/Users/brandon/anaconda/envs/py35/lib/python3.5/site-packages/sklearn/utils/validation.py:42
9: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardSca
ler.
  warnings.warn(msg, _DataConversionWarning)
```

```
Kfold score 1: 0.498677248677
Kfold score 2: 0.718915343915
Kfold score 3: 0.617063492063
Kfold score 4: 0.539021164021
Kfold score 5: 0.569444444444
Kfold score 6: 0.625
Kfold score 7: 0.61044973545
Kfold score 8: 0.638888888889
Kfold score 9: 0.602513227513
Kfold score 10: 0.675925925926
Average score: 0.60958994709
```

The scores had a fairly low variance, indicated that the algorithms were generalizing well and it is robust to unseen data.

The ensembling was done using weighted average score. A multitude of weights of the three models were attempted for a max f1 score.

```
Ensembling...
Best set of weights: [[  3.28277355e-04   9.47608900e-01   5.20628224e-02]]
Corresponding score: 0.670634920635
Output submission file
```

The final ensembled result had a score of 0.72754 on the Kaggle leaderboard.

Justification

The final score is 0.72754, which is higher than the initial score of 0.66. This represent a 10% increase in performance. The final score is still a far far away from the winner's 100% accuracy, meaning there is a lot improvements could be done.

Conclusion

Free-Form Visualization

The figure below is the submission page on Kaggle.

Submission and Description	Private Score	Public Score	Use for Final Score
submission.csv 3 days ago by Brandon *add submission details*	0.72754	0.72754	☐
submission.csv 3 days ago by Brandon *add submission details*	0.72416	0.72416	☐
submission.csv 4 days ago by Brandon *add submission details*	0.72187	0.72187	☐
submission.csv 4 days ago by Brandon *add submission details*	0.66468	0.66468	☐
submission.csv 4 days ago by Brandon *add submission details*	0.66468	0.66468	☐

Reflection

The goal of this project was to predict the forest cover type of test data after training a supervised machine learning models. My approach of solve this problem had three main parts: preprocessing, modeling and ensembling.

The preprocessing included dropping unnecessary features, normalize numerical data. This part requires evaluating the dataset: looking at the skewness of the data, looking at the features of that data.

As for the modeling part, three ML models were chosen. This is an interesting process as I was assuming some models may do good solving this problem. It turned out it is not the case during cross validation. The three chosen models were tuned extensively using a Grid Search Cross Validation technique that optimized for the best f1 score.
The ensembling was key in improving the model performance. It was done by searching for three weights that gave the maximum f1 score. Then ensembled prediction was then rounded up for final evaluation on the Kaggle leaderboard.
The final score of 0.775 showed an 10% improvement over the initial submission. Both Grid Search model tuning and ensembling contributed to this improvement.

Improvement

The biggest area to improve in the future is feature engineering. In my approach, I took in the features from the dataset with minimum feature engineering. Going through some of the Kaggle discussions, it appears that in order to achieve greater prediction accuracy, feature selection seems crucial. There were approaches on Kaggle that added additional features to the dataset for training the model, in order to achieve high score.

Another possible area of improvement will be to look into models not intensively covered in this Nanodegree, such as XGBoost and neural network algorithms. They were not attempted here, but it seems that their proper implementation could very much out-perform the best score obtained.