COMP 47460

Machine Learning

Assignment 2

Conor Murphy

20205251

20/12/2020

Introduction

The following analysis was conducted using a subset of the U.S. Census data, there are 14 features of varying datatypes. The analysis is used to predict weather a person will earn more or less than 50 thousand dollars each year. As this data was used for census analysis, we can assume that it is of good quality and thus no additional cleaning measure are needed. The data also lacks any continuous values, and all values are presented in categorical format meaning there is also no need for the data to be normalised.

Question 1

A) Before conducting the analysis using the KNN, J48 and neural network classification algorithms, some evaluation metrics need to be defined. Firstly, the accuracy is the most obvious of them, this will state how well the model classified the test data. Using just the accuracy measure can be unwise as if the data is overfit to the model this can be a false indicator of performance, to help evaluate this we can use additional metrics such as the true positive rate, the false positive rate, precision, recall and F-measure. The true positive rate and false positive rate can be useful in identifying if a model does a good job between classifying the results of the class they actually belong to. The Precision Recall is the ability for the model to find all the relevant items in the dataset i.e., if 5000 people earn less than 50 thousand the recall would be the ability to identify all the people in that class. Precision is the measure of how accurately it does this i.e., if it just identified everything as the lover class then the recall would be high but the precision low, there is always a trade-off between the two. The F-measure is the harmonic mean between precision and recall, this can be used to indicate a good balance between the two, The higher the F-Measure the better.

Time Complexity is a problem when using very large data set, this was a major issue that I encountered as some models were taking in excessive of 30 minutes to train on the entire dataset. To help reduce this 50% random resampling was carried out, the class distribution pre and post resampling is shown in table 1.1

Class distribution	Full data set	Resampled
>50	4433	2216
<=50K	14126	7063

Table 1.1

Even with the resampling training time was still at an unacceptable level, to further reduce this only 5 fold cross validation will be used for the analysis in this report. Generally, 10 fold cross validation is done, however for the training of neural networks, especially paired with the added complexity of an ensemble, this complexity was simply too much. The function of cross validation remains the same, it will hold out a subset of the data and use this as unseen data once the model is trained, this will allow for an accurate evaluation of the data and it can help to detect overfitting. The performance of the KNN and decision tree were not impacted as much using 10 fold cross validation but to guarantee fairness in the analysis all algorithms will be evaluated using 5 fold cross validation.

The K value of the KNN classifier was set as 1, The j48 algorithm used the default parameters and the Neural network had o hidden layers and had an epoch size of 250 (when the dataset is passed forwards and backward through the dataset once, through forward and backwards propagation), this value was chosen as it produced the most accurate result and had an adequate training time. The K nearest neighbour with k = 1, the J48 decision tree and a neural network algorithm were then run on the dataset.

	KNN	J48	Neural Network
Correct	8037	7943	7944
Accuracy	86.6149	85.6019	85.6127

Table 1.2

Shown above is the number of correctly identified instances by each classification model along with the percentage accuracy of the model. All classifiers do a good job and have high accuracy. The Decision tree and neural network have essential the same accuracy, the nearest neighbour slightly outperforms them with over 86% accuracy, this result was extremely surprising as initially I had started the analysis on the entire data set and realised that the KNN algorithm was the worst performing. I re-ran the algorithm on the full dataset with the KNN algorithm, the results are highlighted in table 1.3, the partial data set was far better performing than the same algorithm with the full dataset, here the 50% resampled subset is clearly outperforming the full dataset, I was also interested to see how different values for k would react and here once again there was a clear discrepancy between the two algorithms, when I ran the full algorithm on the other two classifiers I found that the results were a lot more similar. This shows how the in this instance using the value of 1 on a reduced dataset actually optimised the performance of the KNN classifier.

	full Dataset	Partial Data set
k=1	79.62%	86.6149
k=3	81.8148	83.8884
k=5	82.5691	84.3087

Table 1.3

Class	TP	FP	Precision	Recall	F-Measure
KNN					
>50	0.718	0.087	0.72	0.718	0.719
<=50	0.913	0.282	0.912	0.913	0.912
Weighted Avg	0.866	0.235	0.866	0.866	0.866
J48					
>50	0.592	0.061	0.753	0.592	0.662
<=50	0.939	0.408	0.88	0.939	0.908
Weighted Avg	0.856	0.325	0.85	0.856	0.85
Neural Network					
>50	0.536	0.043	0.795	0.536	0.64
<=50	0.957	0.464	0.868	0.957	0.91
Weighted Avg	0.856	0.364	0.85	0.856	0.846

Table 1.4

Shown in Table 1.3 are the metrics discussed earlier. All classifiers have a good overall weighed mean for precision, recall and the F-measure however it can also be observed that the class >50 performs significantly worse than the other class in these measures. If we look at the J48 and Neural networks, we can see it clearly encounter difficulty identifying the >50 class correctly as both classifiers have a poor True positive rate. The model is assigning a lot of the >50 class to the <=5-class, this can be seen in the false positive rate of the <=50 class. The model is slightly overfit to the <=50 class, this is a result of the class distribution as shown in Figure 1.1, less than

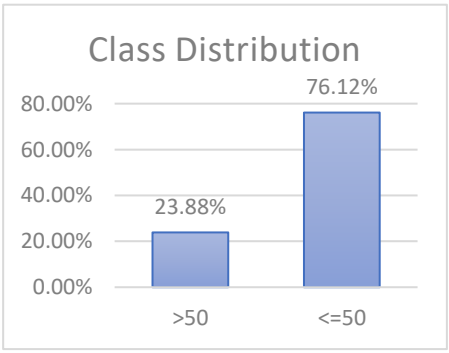

Figure 1.1

a quarter of the data is made up of the >50 class, the overall model accuracy could be Improved with a more evenly distributed training set. In general, the three algorithms certainly do an adequate job with KNN being the best performing but only marginally.

B) An ensemble with bagging was run on the dataset, the size of the ensemble was iteratively increased by 2 in the range 2 to 20. The size of the ensemble essentially means how many different sets will be used, too few and there may not be enough diversity between sets and too many we risk repeating the same patterns over and over again, as the size of the ensemble increases so does the compute time, for example some of the larger ensembles of the Neural network would take over 20 minutes to train, so getting the correct size is important for both accuracy and performance.

	KNN	J48	Neural Network
2	83.4573	85.5696	85.3864
4	85.0738	86.2485	86.0653
6	85.4941	86.4964	86.2485
8	85.7528	87.046	86.3024
10	86.033	87.2616	86.2593
12	86.0006	87.0245	86.3455
14	86.1839	87.046	86.3563
16	86.3886	87.2723	86.2162
18	86.324	87.4124	86.1515
20	86.227	87.3262	86.2916

Table 1.5

Shown above are the results from bagging at different sizes, the best performing models are highlighted in green, unlike part A due to there being over 30 different tests only the accuracy will be used for evaluation as otherwise there would simply be too much data. The best performing size from each class is highlighted in green. The results are also plotted and available in Figure 1.2. Some very interesting observations can be made from the results, the first one is that the decision tree has clearly benefited the most from the bagging, this makes sense as it is an unstable learner so the added diversity in the bags has increased the overall accuracy. The neural network is also an unstable learner but in this instance is not improved as proportionality as the decision tree. The nearest neighbour algorithm actually performed worse in all of the bagging examples than It did in Part A, the reasons for this will be discussed in part D.

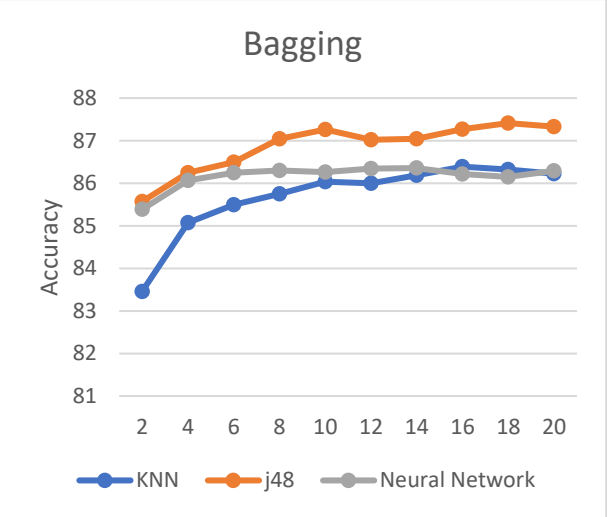

Figure 1.2

Once the best performing size for each classifier was determined then each model's bag size was updated in an attempt to find the optimal model with the highest accuracy. In Weka, the size of the bag is given as a percentage of the overall training set, in 10% increments the size of each model's bag was increased from 10% to 100%. From this I found that the optimal number for each of models was 100% of the bag, from this we can see that each model performed better having more information in each bag which increases diversity. Overall, the bagging ensemble performed very

well and certainly offered a higher accuracy int the case of the j48 and the Neural network classifiers.

C) The same process for part B was adopted however now the random subspace method was implemented. The results are shown in table 1.6, here instead of generating training data with replacement each, set will instead be trained using a smaller subset of the features.

	KNN	J48	Neural Network
2	84.3087	84.2655	84.2224
4	87.6495	85.0846	85.063
6	88.1776	85.1062	85.5265
8	88.8889	85.1385	85.645
10	88.6087	85.4079	85.8498
12	88.8781	85.4726	85.645
14	88.9967	85.4187	85.7743
16	89.1906	85.3001	85.7851
18	89.2876	85.3109	85.6127
20	89.32	85.4618	85.6773

Table 1.6

The best performing models using the random subspace model are highlighted in green. The ensemble of size 20 is the best performing for KNN, however as it was the last set It seemed very possible that the accuracy would increase as the ensemble also increased in size, to test this, I ran the KNN on an ensemble of size 100 and 500 and the results were 89.57% and 89.84% respectively showing that even with an ensemble of 500 members the accuracy is still increasing, however this model did take an excessively long time to train so the .5 percent in accuracy may not be worth the extra compute time. Overall, The KNN was the best performing with nearly 90% accuracy. The J48 and neural network were also very similar with their optimal being roughly 85.5%, showing the KNN drastically outperforming these two. It is also interesting to see the poor performance of the models when the size of the ensemble is 2 as clearly some of the training features are not being included and as a result the accuracy is suffering. Shown in figure 1.3 are the results plotted, again Highlighting just how much better the KNN performs compared to the other two and how with an ensemble size of just 4 it is already considerably better than the classifier in Part A.

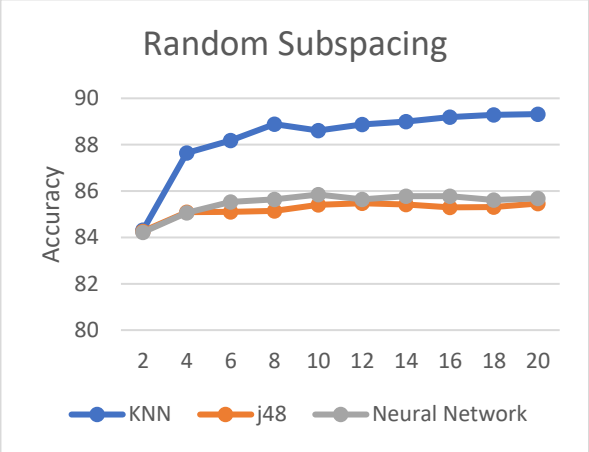

Figure 1.3

The optimal models were then used to see how different numbers of features impact the overall result. In Weka, the Number of features to include is determined as a percentage of the total number of features. Shown are the optimal classifiers with their corresponding percent of features, the best performing classifier using random feature selection is the KNN with an ensemble size of 20 and each item having included 40% of the overall features.

	KNN	J48	Neural Network
Feature number	0.4	0.8	0.9
Accuracy	89.3415	86.7012	86.2162

Table 1.7

D) Based on the analysis conducted in part A, B and C the below results show the model with the highest accuracy from each test. The item “None” in the Ensemble method represents the model created in part A which is not part of any ensemble.

Ensemble Method	KNN	J48	Neural Network
None	86.6149	85.6019	85.6127
Bagging	86.3886	87.4124	86.3563
Random Feature	89.3415	86.7012	86.2162

Table 1.8

The results are exactly in line with the expected outcome from the lecture material. Firstly, take the KNN classifier, this model had the overall best performing classifier of the three initially, this was then extremely interesting to see that the optimised bagging method actually performed worse than the standalone classifier. I was expecting the results to be poor for the bagging of KNN but not worse as this classifier is a stable learner and Bagging tends to work more efficiently on unstable learners with high variance. However as explained in part A the KNN classifier in part A was somewhat of an anomaly and seems to perform very well with k=1 on the resampled data set. I used the full data set to compare and found that the accuracy did increase when using bagging but only by .3% clearly showing that the bagging method is not very effective with the KNN algorithm.

The bagging method did perform as expected on the two unstable learners with the best overall models for these classifiers being generated using this ensemble method. The results were significantly improved with the j48 decision tree by nearly 2%. The difference was far less for the Neural network but there are still some improvements even if they are marginal, in highly complex model these small fraction of percent’s can still be extremely important.

The random feature generation performed exactly as expected from the lecture notes, by adding diversity through different subsets of features it allowed the overall classification accuracy of the stable KNN algorithm to be excellent. From the entire analysis this approach yielded the most accurate model with nearly 90% accuracy. The random feature did add some a level of improved accuracy to the unstable learners but not as much as KNN, this was expected as the random subspace method performs better on models with higher bias.

After concluding the analysis, the best ensemble strategy for each classification algorithm is

- KNN -> Random feature selection of size 20 with 40% of the features in each set
- J48 -> Bagging of size 18 with 100% bag size
- Neural Network -> Bagging of size 14 with 100% bag size

I believe Both the KNN and decision trees to be good approached to take for the census dataset, The KNN had the overall best accuracy with the J48 decision tree not too far behind it. To decide which strategy to use you should consider if you want greater accuracy or speed as the KNN has the best accuracy as a classification model but the j48 is considerably faster at training. I would not recommend using the neural network as part of an ensemble strategy as it was consistently the least accurate and the training times and often took in excess of 30 minutes, and that was only using a subset of data and 5 fold cross validation, so to attempt to do this on a dataset of millions would simply not be feasible. As is always the case with machine learning analysis the accuracy could still be improved with more fine tuning of the model parameters and in this case a more evenly distributed data set would also help improve the accuracy of the model going forward, but the performance of the J48 and KNN classifiers were certainly appropriate for this exercise.

Question 2

A) The interpretability of the simpler supervised learning techniques like KNN, Decision Trees and Linear Regression are somewhat easier to understand that that of neural networks or Ensembles, for example a scatter plot can be drawn out for KNN or the decision tree can easily be illustrated with all of the different nodes and their progression criterial clearly marked just like a flow chart, this can make it far easier to explain how a model came to the conclusion that it did by for example explaining at each node of the tree why it went down the selected path that it did. On the other hand, due to the complex nature of Neural networks and the fact they are essentially a black box, this can make explaining the outcome of a classification extremely difficult and from personal experience If you attempt to explain it in terms of statistics and the maths behind it, the majority of the time the person you are explain it to will be quite confused. Finding the training example that need to be modified or changed in some cases can be easy. If an item in the dataset is clearly an outlier, then it can quite simple be identified and replaced. In KNN if you removed nodes that were close to the new point it may change the classification drastically, however when all of the data is valid and without noise in models like a neural network it can become increasing difficult to understand how each row is impacting the entire model for a particular query. As mentioned earlier, a neural network can be seen as a black box with weighting being contained inside the hidden layers, this makes it difficult to understand how the neural network has assigned weights bases on its cost function. To find out the weight of features in our model we can use metrics like information gain to see how much additional information can be learned from including a certain feature.

B) Bias is the measure of the difference between the average classifier's prediction and the correct value it is trying to predict, so for example if the model were extremely overfit to the training data then the bias would be very low and if it were underfit then the bias would be very high. Variance on the other hand is a measure of error from small changes to the training set. The overfit model with low bias would thus have high variance as if there was a small change to the training data then the outcome would be significantly different. Some algorithms handle bias or variance better than others. Stable learner classification algorithms will generally have a higher bias and lower variance, these include algorithms like Linear regression and KNN. Unstable learners like decision trees and neural networks will generally have a higher variance and a lower bias, thus showing the clear trade-off between the two. Different ensemble strategies can help to reduce the level of bias and variance. For example, using Bagging can help with the variance problems associated with unstable learners, by training classifiers on subsets of data with replacement it can lead to better diversity and thus can help with the overall variance as each classifier is trained on different subsets of the data, this approach will not work as well on the stable learners. Using random feature section can help increase diversity in stable learners. Boosting can be used to address both bias and variance, this approach places and emphasis on incorrectly identified training example in the training process and iteratively improves the quality of the model by address the areas where it performs poorly, this means that the approach will minimise both the bias and variance part of error in the model.

C) There are some drawbacks to using the k-means algorithm for clustering, the first being that we must always specify the number of clusters, which can often lead to the incorrect amount being chosen. The K means algorithm does not handle nonconvex shapes very well and will often split what should clearly be a cluster to the naked eye into two or three separate clusters, this is because the k means algorithm assigns classes based on proximity to a centroid, so the shapes will be spherical. The k means algorithm also does not handle outliers very well, it can also often assign one cluster to only one data point or even will have empty clusters as the value of K increases. K means generally is initialised by selecting random points for the centroids, this can sometime lead to centroids being too

close to each other, other approaches are to use k-means++ which will seek to find clusters than minimise SSE within clusters or to use farthest first which will spread centroid out as far as possible from each other at initialisation.

D) In linear regression R^2 is a measure to see how close the data is fitted to the line; it essentially measures the goodness of the fit of a linear model. R^2 is also known as the correlation of determination and can be calculated with the equation SSR/SST. R^2 can inform us how close the regression line actually matches the real data. R^2 is measured between 0 and 1, 0 indicating a very poor fit with 1 being the best possible fit. For example, if a model had a 0.9 R^2 value then we could say there is a strong fit and that it accounts for 90% of the variation in the linear model, the higher the variance that is accounted for the better. There are some drawbacks to R^2, for one it can not tell you if the model is biased, for this you would need to look at the residual plots. if the linear model contains more multiple independent variables then then using the Adjusted R^2 Value is wiser.

E) Precision and recall are important metrics for evaluating machine learning models. Recall is the model's ability to retrieve all of the relevant results from the dataset , whereas precision is the measure of how accurately the relevant results were retrieved. A good way to understand them is with an example, if you had dataset with a binary classifier, where 100 of the classes were "yes" , the recall would be how many of these classes were actually classified as yes, this can lead to problems as if the model simply classified everything as yes then the recall would still be 1. The precision is the amount of the classes that were classified as yes that are actually yes. Naturally, we would want the highest level of both precision and recall but this is often not possible due to the trade-off between the two metrics. Generally, with a higher recall this will result in a lower precision and vice versa. In many instances the context of what the model is being used for can determine if precision or recall should be prioritised, for example at airport security you would want a far higher recall as if one dangerous person slips thought the results could be catastrophic and the drop in precision will only result in a few more people being inconveniently searched. On the other hand, if there was model for email spam detection here precision is more important as if an email is classified as spam and it was important then you may miss it. A good way to determine what level of precision and recall is to plot them on a precision recall curve. The F1-measure can also be useful as it the harmonic mean of precision and recall, which measure the balance between the two.

F) The training of neural networks takes place in multiple steps. To understand how parameters are learned it is helpful to understand the components that make up the neural network, the input layer is the first layer and will take the values to be used for training this could be the pixels of an image, these values are then passed into the hidden layers which will multiply each input layer by its weight and compare it against an activation function, this is generally the Sigmoid function, but others can also be used. Some level of bias can also be added to each node in the hidden layers, initially these values are guesses but they will be updated in a later stage. If the activation threshold is met then this node will progress to the next level and the process is repeated until the output layer is reached, here the output feature with the highest activation threshold will be adopted. This is only the first passthrough and it is called forward propagation, now the cost function of the outputs can be generated. Using a loss function optimisation algorithm like gradient descent or stochastic gradient descent we can then propagate backwards thought the network and adjust bias and weight values on each neuron until we reach the local minimum of the loss function. The cycle of forward and back propagation will eventually result in a highly accurate trained neural network; however, this training time can be very costly and as the steps are performed iteratively on the hidden layer it can mean that neural networks are a black box as we do not know how they arrived at the values for each node in the network. To summarise the parameters are learned using a gradient descent algorithm to find the bias and weight values that minimise the cost function in a neural network.