Design and Application of a Machine Learning System for a Practical Problem

Submitted as part of the requirements for:

CE802 Machine Learning and Data Mining

**Name**: Peter Gent

**Date**: 14 January 2019

Reg no: 1802559

**Abstract**. This sample document is not by any means complete and accurate. It merely illustrates some of the basic mechanisms by which Microsoft Word 2007 can be used to write technical reports. Guidelines about the structure and contents of each report will be distributed separately.

**Keywords**: Keywords help your reader to set the context to the report. Name the relevant disciplines and fields of research, such as: software design, computer networks, circuit theory.

Pilot study proposal 3

Comparative study: 4

DecisionTree: 4

SVM: 5

KNN : 6

Appendix: 7

# Pilot study proposal

The problem presented by the restaurant chain is to try and classify potential new business locations for the chain into unsuccessful and successful businesses by looking at a feature space. This is a classification problem rather than a regression problem as the manager wants to identify potential business locations as being indicators of successful businesses or not. From this point there are several different approaches that could be taken:The best way of classifying such business ventures would be with a soft margin SVM over a multivariate feature space. The expected feature space could include features such as:

1. proximity to nearest restaurant
2. proximity to nearest restaurant owned by chain
3. number of restaurants in neighbourhood of same cuisine.
4. affluence per capita of people in the neighbourhood.
5. footfall of potential customers in area as a potential indicator of success

The feature space could have included profitability but this would outweigh any of the subtle links between characteristics of the neighbourhood that are sought after when data mining, thus this is better left as an indicator of success or not as a label for supervised learning. For the most part the features in the proposed feature space revolve around the affluence of people and the amount of competition in a neighbourhood. A high number of restaurants in the neighbourhood indicates that it would be a very competitive market, and with restaurants in the same chain as it could be harmful to have two restaurants competing for the same customer base. likewise in terms of location it could be important to know the footfall or how many potential customers walk by the restaurant space and the average affluence or free cash on average per capita to spend, indicating how much a person is willing to spend in a given chain.

Once the solution has been trained using supervised learning, the success or failure of the model could then be tested against a validation or test set to prove that it would work in the wild. There are also a number of alternative ways that the model can be judged to be successful or unsuccessful such as accuracy, which is often dependent on the quality and quantity of the training set. By using stratification (folding the data) it is possible to ensure that each fold of data has the same proportions of results as the previous layer and then each fold can be tested against to cross validate the learning procedure. That way with a ten fold test set the successfulness of a machine learning procedure can be graded out of ten. Additionally other forms of criterion can be applied such as a confusion matrix could also be used to identify number of true positive or negatives and false positives and false negatives. The Kappa statistic can be derived from the confusion matrix to work out how much better than random guessing the solution provides.

# Comparative study:

In this study several kinds of ai techniques will be covered, these are: DecisionTrees, SVM and KNN. As there was no built in way to prune trees the next best solution was to either create a pruning interface or to limit the number of leaf nodes to scale the amount of overfitting in the solution. It is because of time constraints that this action was taken.

On the other hand folding of the data was achieved through cross validation itself, as the test also trains the model against the data.

### DecisionTree:

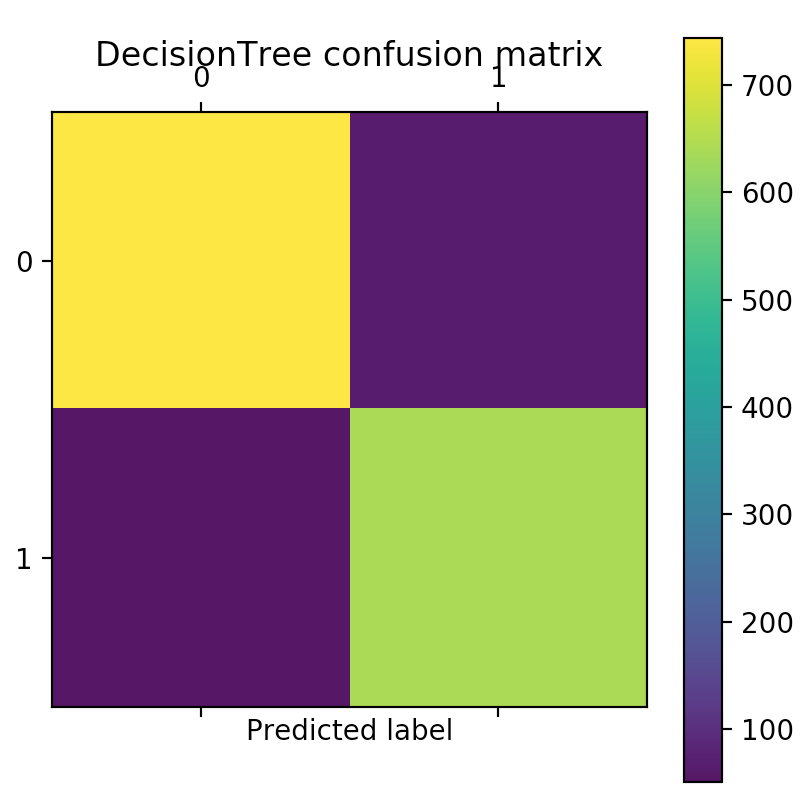


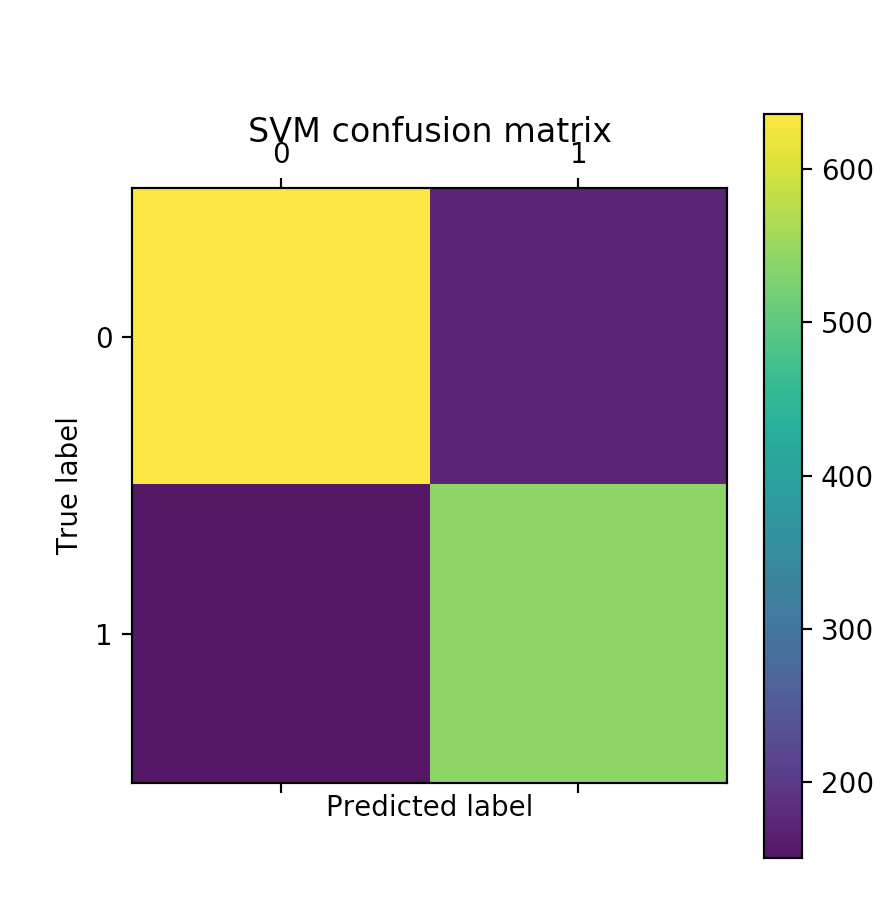
Figure 1: confusion matrix for DecisionTree

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  | Mean |
| Cross validation | 0.84666667 | 0.78 | 0.79333333 | 0.80666667 | 0.74 | 0.74666667 | 0.78 | 0.82666667 | 0.75333333 | 0.73333333 | 0.780666667 |
| Kappa score | 0.8445998445998446 |  |  |  |  |  |  |  |  |  |  |

The decision tree solution has by a marked degree better performance with a mean cross validation score of 0.78 vs the 0.64 rating of the SVM and the 0.62 score for KNN solutions.It also exhibits a very performance in the confusion matrix with fewer than one hundred results being classified as false positives or false negatives. Finally, the kappa statistic of 0.8445 is also very high making the selection of successful and unsuccessful restaurants very unlikely to be selected by chance. It is because of this top performance rating for decision trees that the SVM approach or KNN approach was dropped in favour of a decision tree approach for the final prediction.

The code for the solution is pretty straight forward (see appendix I).

### SVM:

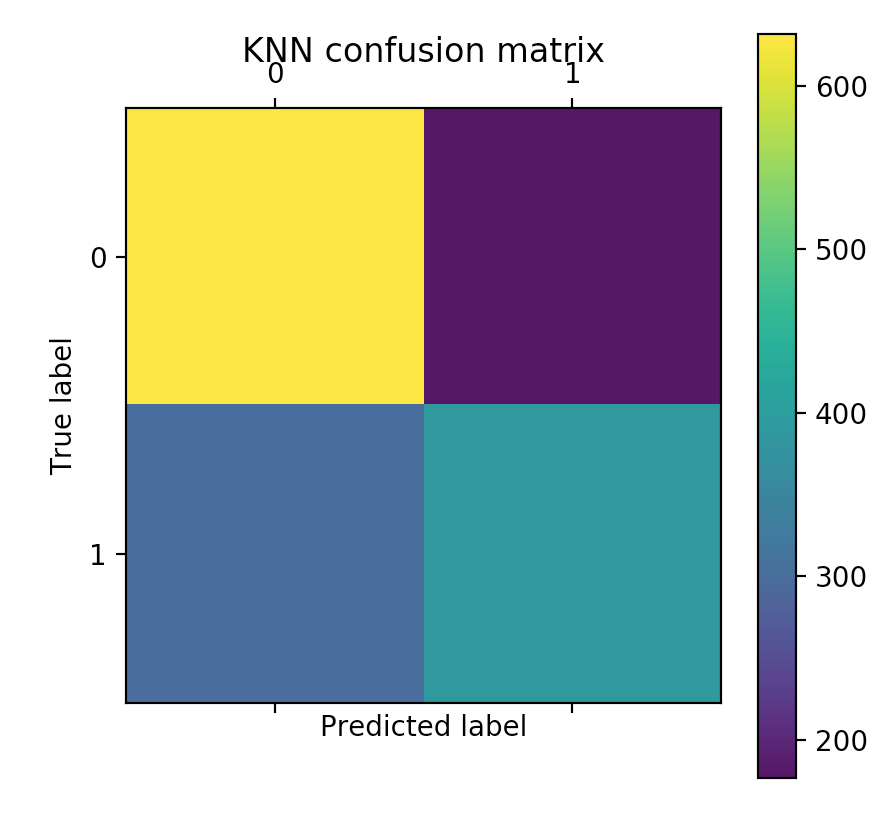


|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  | Mean |
| Cross validation | 0.66666667 | 0.67333333 | 0.66666667 | 0.7 | 0.61333333 | 0.70666667 | 0.61333333 | 0.62 | 0.57333333 | 0.57333333 | 0.640666666 |
| Kappa score | 0.566316924169711 |  |  |  |  |  |  |  |  |  |  |

The SVM solution provides a performance that is somewhat better than that of KNN and comparable to that of the DecisionTreeClassifier. The Kappa score of 0.56 means that the solution identifies the right results slightly more than 50% of the time meaning that it has slightly better performance compared with guessing. The SVM also has a relatively low number of false positives or false negatives ranging in the less than 200 results being located in one of these subdivisions.

The code for the solution is pretty straight forward (see appendix I).

### KNN :

cvs:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  | Mean |
| Cross validation | 0.64666667 | 0.62666667 | 0.64666667 | 0.70666667 | 0.58 | 0.66666667 | 0.59333333 | 0.61333333 | 0.54666667 | 0.64 | 0.626666668 |
| Kappa score | 0.3516226198977096 |  |  |  |  |  |  |  |  |  |  |

The KNN performance is by some considerable amount the worst performance of the three. This results in a confusion matrix that is by the standards of SVMs and DecisionTrees far inferior, with more false positives. This observation on false positives is signified by a lighter shade of blue rather than purple for one of the quartiles.

The code for the solution is pretty straight forward (see appendix I).

## Appendix:

[1] code for solution

**from** sklearn.model\_selection **import** KFold, cross\_val\_score

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn **import** tree

**from** sklearn **import** svm

**from** sklearn **import** neighbors

**from** sklearn.model\_selection **import** KFold, cross\_val\_score

**from** sklearn.metrics **import** cohen\_kappa\_score, confusion\_matrix

**from** sklearn.model\_selection **import** StratifiedKFold

**import** pandas **as** pd

**import** graphviz

**import** matplotlib.pyplot **as** plt

**class** DecisionTree:

*# pruned tree classifyer*

**def** \_\_init\_\_(self, train\_data,train\_class\_data):

k\_fold = KFold(n\_splits=10)

self.dt = DecisionTreeClassifier(max\_leaf\_nodes=80, criterion=**"entropy"**)

cross\_val\_score\_tree = cross\_val\_score(self.dt, train\_data, train\_class\_data, cv=k\_fold, n\_jobs=-1)

self.dt.fit(train\_data,train\_class\_data)

*#print(cross\_val\_score\_tree)*

**def** visualiseTree(self):

exported\_tree = tree.export\_graphviz(self.dt,out\_file=**None**)

exported\_graph = graphviz.Source(exported\_tree)

exported\_graph.render(**"Restaurants"**)

**def** predict(self,test\_data):

**return** self.dt.predict(test\_data)

**def** scoreTree(self, training\_data, training\_class\_data):

k\_fold = KFold(n\_splits=10)

cm = confusion\_matrix(training\_class\_data,self.dt.predict(training\_data))

kappa = cohen\_kappa\_score(training\_class\_data,self.dt.predict(training\_data))

print(**"Statistics for decision tree implementation: "**)

print(cross\_val\_score\_tree)

print(kappa)

plt.matshow(cm)

plt.title(**'DecisionTree confusion matrix'**)

plt.colorbar()

plt.ylabel(**'True label'**)

plt.xlabel(**'Predicted label'**)

plt.show()

**class** SVM:

**def** \_\_init\_\_(self,train\_data,train\_class\_data):

self.\_svm = svm.SVC(gamma=**'scale'**,decision\_function\_shape=**'ovo'**)

self.dt.fit(train\_data, train\_class\_data)

cvs = cross\_val\_score(self.\_svm, training\_data, training\_class\_data, cv=k\_fold, n\_jobs=-1)

print(**"SVM cvs: "**)

print(cvs)

**def** scoreSVM(self,training\_data,training\_class\_data):

print(**"SVM statistics:"**)

k\_fold = KFold(n\_splits=10)

cm = confusion\_matrix(training\_class\_data,self.\_svm.predict(training\_data))

cks = cohen\_kappa\_score(training\_class\_data,self.\_svm.predict(training\_data))

print(**"cvs:"**)

print(cvs)

print(**"cks: "**)

print(cks)

plt.matshow(cm)

plt.title(**'SVM confusion matrix'**)

plt.colorbar()

plt.ylabel(**'True label'**)

plt.xlabel(**'Predicted label'**)

plt.show()

*## https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/*

**class** KNN:

**def** \_\_init\_\_(self,train\_data,train\_class\_data):

self.\_\_knn = neighbors.KNeighborsClassifier(n\_neighbors=14)

k\_fold = KFold(n\_splits=10)

self.dt.fit(train\_data, train\_class\_data)

cvs = cross\_val\_score(self.\_svm, training\_data, training\_class\_data, cv=k\_fold, n\_jobs=-1)

print(**"SVM cvs: "**)

print(cvs)

**def** scoreNN(self,training\_data,training\_class\_data):

cks = cohen\_kappa\_score(training\_class\_data,self.\_\_knn.predict(training\_data))

cm = confusion\_matrix(training\_class\_data,self.\_\_knn.predict(training\_data))

print(**'cvs:'**)

print(cvs)

print(**"cks:"**)

print(cks)

plt.matshow(cm)

plt.title(**'KNN confusion matrix'**)

plt.colorbar()

plt.ylabel(**'True label'**)

plt.xlabel(**'Predicted label'**)

plt.show()

**def** seperateData(training\_file):

regular\_training\_data = training\_file.iloc[:,:-1]

training\_class\_data = training\_file.iloc[:,-1:]

**return** regular\_training\_data,training\_class\_data

**def** main():

test\_filename = **"CE802\_Ass\_2018\_Test.csv"**

training\_filename = **"CE802\_Ass\_2018\_Data.csv"**

train\_df = pd.read\_csv(training\_filename)

test\_df = pd.read\_csv(test\_filename)

training\_data, training\_shop\_class = seperateData(train\_df)

test\_data, test\_shop\_class = seperateData(test\_df)

*#create and prune tree followed by scoring.*

udt = DecisionTree(training\_data,training\_shop\_class)

*#udt.visualiseTree()*

*#udt.scoreTree(training\_data,training\_shop\_class)*

write\_file = open(**"predictions.csv"**,**'w'**)

**for** prediction **in** udt.predict(test\_data):

write\_file.write(str(prediction) + **' \n'**)

write\_file.close()

*#support\_vector\_machine = SVM(training\_data,training\_shop\_class)*

*#support\_vector\_machine.scoreSVM(training\_data,training\_shop\_class)*

*#knn = KNN(training\_data,training\_shop\_class)*

*#knn.scoreNN(training\_data,training\_shop\_class)*

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

main()