Fatec Franca Dr. Thomaz Novelino
27/09/2024

Disciplina Aprendizagem de Máquina
Docente Jaqueline Brigladori Pugliesi
Discente Samuel Luiz Martins dos Santos
Registro 1091392223033

Análise de algoritmos geradores de regras e árvores no WEKA

SUMÁRIO

***

Introdução

Este trabalho tem como intuito analisar, de forma comparativa, os resultados de diferentes algoritmos na aprendizado de máquina pelo programa WEKA, um software de código aberto distribuído sob a Licença Pública Geral GNU.

O dataset utilizado para essa análise chama-se Car Evaluation e download foi feito através da página UC Irvine Machine Learning Repository.

As respostas completas retornadas pelo programa WEKA encontram-se na sessão “**apêndice**” deste documento.

Características do Conjunto de Dados

Nome: Car Evaluation Database

Descrição: Derivado de um modelo hierárquico simples de decisão, este banco de dados pode ser útil para testar métodos de indução construtiva e descoberta de estrutura.

Tipo: Multivariado

Área de Aplicação: Outros

Tarefas Associadas: Classificação

Tipo de Atributo: Categórico

Número de Instâncias: 1728

Número de Atributos: 6

Nota: Não possui dados faltantes.

O Car Evaluation Database foi derivado de um modelo hierárquico simples de decisão desenvolvido originalmente para a demonstração do DEX, conforme descrito no trabalho de M. Bohanec e V. Rajkovic: "Expert system for decision making," publicado na Sistemica 1(1), pp. 145-157, 1990.

O modelo avalia carros de acordo com a seguinte estrutura de conceitos:

- **CAR**: Aceitabilidade do carro
    - **PRICE**: Preço geral
        - **buying**: Preço de compra. Categorizado como:
            - **vhigh** (muito alto);
            - **high** (alto);
            - **med** (médio);
            - **low** (baixo).
        - **maint**: Custo de manutenção. Categorizado como:
            - **vhigh** (muito alto);
            - **high** (alto);
            - **med** (médio);

                - **low** (baixo).
        - **TECH**: Características técnicas
            - **COMFORT**: Conforto
                - **doors**: Número de portas. Categorizado como:
                    - **2**;
                    - **3**;
                    - **4**;
                    - **5more** (5 ou mais).
                - **persons**: Capacidade de pessoas. Categorizado como:
                    - **2**;
                    - **4**;
                    - **more** (mais).
                - **lug_boot**: Tamanho do porta-malas. Categorizado como:
                    - **small** (pequeno);
                    - **med** (médio);
                    - **big** (grande).
            - **safety**: Segurança estimada do carro. Categorizado como:
                - **low**;
                - **med**;
                - **high**.

Os **Rótulos de Classe** são escritos da seguinte maneira:

- **unacc**: Inaceitável
- **acc**: Aceitável
- **good**: Bom
- **vgood**: Muito bom

Os atributos de entrada estão escritos em minúsculas. Além do conceito alvo (CAR), o modelo inclui três conceitos intermediários: PRICE, TECH, e COMFORT. Cada conceito no modelo original está relacionado aos seus descendentes de nível inferior por um conjunto de exemplos.

O Car Evaluation Database contém exemplos com a informação estrutural removida, ou seja, relaciona diretamente CAR aos seis atributos de entrada: buying, maint, doors, persons, lug_boot, safety.

Escolhendo algoritmos de regras

Essa etapa possui o intuito de executar cada algoritmo de regra oferecido pelo WEKA e selecionar os mais compreensíveis ao discente, apontando aspectos que tiveram influência na tomada da decisão.

Algoritmo	Experiência	Detalhamento
DecisionTable	Facilmente compreensível	Possui uma estrutura de exibição de resposta clara composta por: ● Informações da execução ● Modelo do classificador ● Uma validação cruzada estratificada ● Detalhes de precisão por classe ● E uma matriz de confusão
JRip	Facilmente compreensível	Possui exatamente a mesma estrutura de **DecisionTable**, porém com uma enorme diferença no Modelo classificador, apresentando linhas corridas dos atributos que geraram o resultado de classificação.
M5Rules	Desativado	Infelizmente não foi possível executar esse algoritmo.
OneR	Facilmente compreensível	Também possui a mesma estrutura que **DecisionTable** e **JRip**, mas retornou uma resposta bem mais simplificada que os dois, o que passou a impressão de ser menos sofisticado na análise, principalmente pela alta taxa de erros gerados.
PART	Facilmente compreensível	Possui a estrutura já mencionada anteriormente e uma semelhança com **JRip** no modelo do classificador, porém em um formato mais semelhante com tags ao invés de linhas corridas.
ZeroR	Facilmente compreensível	Possui a estrutura já mencionada anteriormente e uma semelhança com **OneR** na simplicidade, mas aparentemente ele só prevê a classe mais frequente, independentemente dos atributos.

Após a execução e análise de cada algoritmo de regra, foram escolhidos **DecisionTable** e **PART** para seguir com as análises. Ambos parecem ser bem concisos e bem elaborados para tarefas de classificação complexas, em especial PART, por retornar detalhamento mais aprofundado na parte de classificação de modelo, além de surpreendentemente ter retornado a maior porcentagem de acertos.

As respostas de suas execuções se encontram nos apêndices **A (DecisionTable)** e **B (PART)**.

Comparação entre DecisionTable e PART

Toda a estrutura é exatamente igual, exceto na sessão “Classifier model” onde o algoritmo PART tende a ser mais detalhado com os valores de instâncias utilizados para fazer a lista de decisão, mas mais complexo de compreender sem um conhecimento prévio em comparação com o modelo de DecisionTable.

Podemos notar que o número de regras utilizado pela DecisionTable (432) foi muito maior que a de PART (68) e, por decorrência, PART teve um tempo de construção (0.03 segundos) quatro vezes menor que DecisionTable (0.16 segundos).

No processo de construção de cada modelo, o DecisionTable é uma abordagem que busca criar uma tabela de decisões baseada em subconjuntos de atributos, o que resulta em um maior número de regras para cobrir a maior variedade de combinações possíveis de atributos. Já o PART é um algoritmo de indução de regras que gera uma lista de regras de decisão criando regras incrementais a partir de subárvores podadas[1] (de uma árvore de decisão).

DecisionTable	PART
Decision Table: Number of training instances: 1728 Number of Rules : 432 Non matches covered by Majority class. Best first. Start set: no attributes Search direction: forward Stale search after 5 node expansions Total number of subsets evaluated: 22 Merit of best subset found: 94.329 Evaluation (for feature selection): CV (leave one out) Feature set: 1,2,4,5,6,7 Time taken to build model: 0.16 seconds	PART decision list ---------- safety = low: unacc (576.0) persons = 2: unacc (384.0) buying = vhigh AND maint = vhigh: unacc (48.0) buying = high AND maint = vhigh: unacc (48.0) safety = med AND lug_boot = small AND buying = vhigh: unacc (24.0) safety = med AND lug_boot = small AND buying = high: unacc (24.0) ***[...] (recorte. Você pode encontrar a resposta completa do algoritmo no apêndice B)*** persons = 4: acc (2.0) : good (2.0) Number of Rules : 68 Time taken to build model: 0.03 seconds

[1] O termo "podada" significa que ramos desnecessários, que não trazem ganho significativo de informação, foram removidos para simplificar a árvore.

Nas porcentagem de acertos, PART apresentou um melhor resultado apesar de ter criado menos regras de decisão:

DecisionTable		PART	
Correctly Classified Instances	1573	Correctly Classified Instances	1655
91.0301 %		95.7755 %	
Incorrectly Classified Instances	155	Incorrectly Classified Instances	73
8.9699 %		4.2245 %	

Comparando as matrizes de confusão entre os algoritmos, podemos notar que os atributos “inaceitáveis” (unacc) e “aceitáveis” (acc) tiveram uma margem parecida, mas DesicionTable obteve uma taxa de confusão maior no atributo “aceitável” o associando mais ao inaceitável. E em relação aos demais atributos, “bom” (good) e “muito bom” (vgood), o algoritmo PART obteve uma análise muito mais correta em relação ao DesicionTable:

DecisionTable

a	b	c	d	<-- classified as
1173	34	3	0	| a = unacc
65	308	9	2	| b = acc
8	10	45	6	| c = good
2	5	11	47	| d = vgood

PART

a	b	c	d	<-- classified as
1180	26	4	0	| a = unacc
6	360	16	2	| b = acc
0	15	51	3	| c = good
0	1	0	64	| d = vgood

Escolhendo algoritmos de árvore

Esse trecho repete os mesmos processos de análise feitos na escolha dos algoritmos de regra, mas com um preconceito em relação às árvores e em como elas são utilizadas na estrutura de dados ao exibir seus relacionamentos. Em outras palavras, aqueles algoritmos que não apresentaram uma estrutura visual baseada em ramificações, foi automaticamente considerado difícil de se compreender.

Também é importante notar que a estrutura de exibição das respostas destes algoritmos feitos pelo WEKA, assemelha-se à dos algoritmos de regra, portanto o foco da análise mira a sessão **Modelo de Classificação**. Não só isso, mas a maioria das respostas utilizando estes algoritmos tiveram uma excelente porcentagem de acerto em comparação com os algoritmos de regra.

Algoritmo	Experiência	Detalhamento
DecisionStump	Incompreensível	Este algoritmo em especial aparentou ser mais simples e portanto teve uma taxa de erros muito elevada.

HoeffdingTree	Incompreensível	Não consegui compreender muito bem, mas sua taxa de acertos foi boa.
J48	Facilmente compreensível	Retornou de forma muito bem organizada e detalhada uma árvore que exibe os filhos das entidades incluindo os atributos utilizados para a classificação.
LMT	Dificilmente compreensível	Esse algoritmo aparenta trazer uma exibição visual muito detalhada de uma árvore, porém em uma estrutura não muito comum. O mais surpreendente foi sua taxa de acerto que quase chegou a 100%.
M5P	Desativado	Infelizmente não foi possível executar esse algoritmo.
RandomForest	Incompreensível	Resultou em uma boa taxa de acertos, mas não apresentou nenhuma forma visual de árvore.
RandomTree	Facilmente compreensível	Apresentou um formato bastante semelhante ao **J48**, mas com uma taxa de acertos um pouco inferior.
REPTree	Facilmente compreensível	Apresentou um formato bastante semelhante ao **J48** e **RandomTree**, mas com uma taxa de acertos um pouco inferior ao **J48**.

Após a execução e análise de cada algoritmo de árvore, decidi seguir com **J48**, por ter apresentado uma imagem clara da estrutura da árvore e uma boa taxa de acerto e o algoritmo **LMT** apesar de sua leitura ser mais complexa, pois seu resultado de acertos foi impressionante. Ambos também parecem ser bem concisos e adequados para tarefas de classificação complexas, especialmente o LMT.

As respostas de suas execuções se encontram nos apêndices **C (J48)** e **D (LMT)**.

Comparação entre **J48** e **LMT**

Não convém exibir os modelos de classe de ambos os algoritmos devido aos seus tamanhos. É possível encontrá-los nos anexos referidos anteriormente.

O J48 divide o conjunto de dados em várias partes usando os atributos mais relevantes. Ele faz isso calculando o ganho de informação para cada atributo, escolhendo aquele que melhor separa as instâncias no conjunto de treinamento. O processo de **poda** remove subárvores que têm baixo poder preditivo para evitar o "overfitting" (superajuste), o que torna o modelo mais generalizável. Ele retornou uma estrutura visual de suas ramificações juntamente com o número de instâncias compatíveis com cada regra, o seu tempo de construção foi zero segundos, o tamanho da árvore foi de 182 e a quantidade de folha foi equivalente a 131.

O LMT demorou 1.92 segundos para ser construído, maior que o J48 devido ao cálculo das equações logísticas em cada folha, o tamanho da árvore foi 21 e o número de folhas igual a 15, ambos muito menores que o J48. O LMT primeiro constroi uma árvore de decisão básica, como o J48, mas em vez de parar em cada folha, ele ajusta e exibe um **modelo de regressão logística** para cada folha da árvore, ou seja, ao invés de simplesmente atribuir uma classe, ele fornece uma **equação logística** para calcular a probabilidade de uma instância pertencer a uma classe, permitindo que o LMT modele relações não lineares o que pode ser útil em problemas onde o resultado não é estritamente binário. Isso torna o LMT mais robusto em termos de capacidade de modelar classes complexas, embora mais lento para construir e mais difícil de interpretar. Ele exibiu a árvore, mas cada folha é apresentada juntamente com um modelo logístico que foi ajustado para uma determinada quantidade de instâncias, com um indicador do erro de ajuste e, para cada folha, há equações que começam com um valor base, seguido por coeficientes multiplicados pelos atributos correspondentes. O que significa que uma probabilidade de uma classe ser específica é calculada com base nestes coeficientes para diferentes valores de um determinado atributo.

Em relação à porcentagem de acerto, ambos foram muito bons, mas LMT obteve a melhor pontuação.

J48		**LMT**	
Correctly Classified Instances 92.3611 %	1596	Correctly Classified Instances 98.7847 %	1707
Incorrectly Classified Instances 7.6389 %	132	Incorrectly Classified Instances 1.2153 %	21

Comparando as matrizes de confusão entre os algoritmos, podemos notar que o algoritmo **J48** obteve um número de confusões relevantes entre os atributos atributos “inaceitáveis” (unacc) e “aceitáveis” (acc) e o algoritmo **LMT** obteve resultados incríveis em todos os atributos. Mesmo tendo havido alguns erros, sua quantidade é quase irrelevante:

J48					**LMT**				
a	b	c	d	<-- classified as	a	b	c	d	<-- classified as
1164	43	3	0	| a = unacc	1207	3	0	0	| a = unacc
33	333	11	7	| b = acc	4	371	7	2	| b = acc
0	17	42	10	| c = good	0	4	65	0	| c = good
0	3	5	57	| d = vgood	0	1	0	64	| d = vgood

Conclusão e comparação entre PART e LMT

Em resumo, **Algoritmos de Regras** criam um conjunto de regras para classificar dados. Cada regra é uma condição que, se satisfeita, leva a uma conclusão sobre a classe. Uma regra pode ser: "***Se buying=high e safety=low, então a classe é não aceitável.***"

As regras são fáceis de entender e interpretar. Cada decisão é explícita e pode ser verificada. Elas são uma decisão binária sobre a classe com base em uma condição específica tornando as coisa mais simples e diretas, mas podem ser limitadas se o conjunto de dados for complexo, especialmente se as regras não forem bem combinadas.

Já os **Algoritmos de Árvore** dividem os dados em subgrupos com base em atributos, construindo uma estrutura semelhante às ramificações de uma árvore. Cada nó interno da árvore representa uma decisão com base em um atributo, e cada folha representa uma classe.

A árvore é construída de forma hierárquica, dividindo o conjunto de dados em diferentes níveis. Cada nível representa uma decisão adicional sobre um atributo e podem capturar interações complexas entre esse atributos, por exemplo, uma árvore pode primeiro verificar ***safety***, e depois dividir com base em ***buying*** e ***persons***, capturando interações entre esses atributos.

As árvores podem ser ajustadas para incluir ou excluir atributos. Isso pode levar a um melhor desempenho em conjuntos de dados onde a relação entre atributos não é linear ou simples. Por isso muitas vezes têm uma taxa de acerto mais alta porque podem ajustar-se melhor às complexidades dos dados.

Para finalizar, gostaria de comparar a taxa de acertos e a matriz de confusão dentre os algoritmos de regra e de árvore que obtiveram a maior quantidade de respostas corretas, relevando o fato de algum algoritmo de regra utilizar conceitos de árvore em suas classificações ou algum algoritmo de árvore utilizar cálculos probabilísticos muito complexos para serem adaptados a um algoritmo de regra.

Porcentagem de acerto. O **LMT** foi o melhor:

PART		**LMT**	
Correctly Classified Instances 95.7755 %	1655	Correctly Classified Instances 98.7847 %	1707
Incorrectly Classified Instances 4.2245 %	73	Incorrectly Classified Instances 1.2153 %	21

Comparando as matrizes de confusão entre os algoritmos, podemos notar que ambos compararam de forma exatamente igual o atributo “muito bom” (vgoogd) e de forma muito próxima o atributo “aceitável” (acc). Mas nos demais atributos **LMT** entrega um resultado melhor:

PART					**LMT**				
a	b	c	d	<-- classified as	a	b	c	d	<-- classified as
1180	26	4	0	| a = unacc	1207	3	0	0	| a = unacc
6	360	16	2	| b = acc	4	371	7	2	| b = acc
0	15	51	3	| c = good	0	4	65	0	| c = good
0	1	0	64	| d = vgood	0	1	0	64	| d = vgood

***

Referências

Página web do WECA: https://waikato.github.io/weka-wiki/

Repositório de datasets: http://archive.ics.uci.edu/

Dataset utilizado na pesquisa:
http://archive.ics.uci.edu/dataset/19/car+evaluation

Trabalho de M. Bohanec e V. Rajkovic: "Expert system for decision making," publicado na Sistemica 1(1), pp. 145-157, 1990. Através do link: https://repositorio.ufscar.br/bitstream/handle/ufscar/366/1698.pdf?sequence=1&isAllowed=y

Apêndices

Apêndice A

```
DecisionTable
=== Run information ===

Scheme:       weka.classifiers.rules.DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N
5"
Relation:     car
Instances:    1728
Attributes:   7
              buying
              maint
              doors
              persons
              lug_boot
              safety
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Decision Table:

Number of training instances: 1728
Number of Rules : 432
Non matches covered by Majority class.
              Best first.
              Start set: no attributes
              Search direction: forward
              Stale search after 5 node expansions
              Total number of subsets evaluated: 22
              Merit of best subset found:   94.329
Evaluation (for feature selection): CV (leave one out)
```

```
Feature set: 1,2,4,5,6,7

Time taken to build model: 0.16 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1573               91.0301 %
Incorrectly Classified Instances       155                8.9699 %
Kappa statistic                          0.7987
Mean absolute error                      0.2748
Root mean squared error                  0.322
Relative absolute error                119.9872 %
Root relative squared error             95.2225 %
Total Number of Instances             1728

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,969    0,145    0,940      0,969    0,954      0,844    0,978     0,989     unacc
               0,802    0,036    0,863      0,802    0,831      0,786    0,967     0,869     acc
               0,652    0,014    0,662      0,652    0,657      0,643    0,941     0,654     good
               0,723    0,005    0,855      0,723    0,783      0,779    0,965     0,796     vgood
Weighted Avg.  0,910    0,110    0,908      0,910    0,909      0,820    0,973     0,941

=== Confusion Matrix ===

   a    b    c    d   <-- classified as
 1173   34    3    0 |    a = unacc
  65  308    9    2 |    b = acc
   8   10   45    6 |    c = good
   2    5   11   47 |    d = vgood
```

Apêndice B

```
PART
=== Run information ===

Scheme:       weka.classifiers.rules.PART -C 0.25 -M 2
Relation:     car
Instances:    1728
Attributes:   7
              buying
              maint
              doors
              persons
              lug_boot
              safety
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

PART decision list
------------------

safety = low: unacc (576.0)
```

```
persons = 2: unacc (384.0)

buying = vhigh AND
maint = vhigh: unacc (48.0)

buying = high AND
maint = vhigh: unacc (48.0)

safety = med AND
lug_boot = small AND
buying = vhigh: unacc (24.0)

safety = med AND
lug_boot = small AND
buying = high: unacc (24.0)

buying = high AND
doors = 4: acc (30.0)

safety = med AND
maint = high AND
buying = low: acc (24.0/1.0)

buying = high AND
lug_boot = big: acc (36.0)

safety = med AND
maint = high AND
buying = vhigh: unacc (16.0)

safety = med AND
lug_boot = big AND
buying = vhigh: acc (16.0)

safety = med AND
lug_boot = big AND
maint = vhigh: acc (16.0)

safety = med AND
lug_boot = big AND
buying = low: good (16.0)

maint = vhigh AND
safety = high AND
doors = 3: acc (12.0)

buying = high AND
safety = high AND
doors = 3: acc (12.0)

safety = med AND
maint = vhigh AND
lug_boot = small: unacc (16.0)

buying = high AND
doors = 5more: acc (18.0)

buying = vhigh AND
```

```
maint = med: acc (32.0/4.0)

buying = vhigh AND
maint = low: acc (32.0/4.0)

buying = vhigh: unacc (24.0)

safety = med AND
maint = vhigh AND
doors = 2: unacc (4.0)

safety = med AND
maint = vhigh AND
doors = 4: acc (4.0)

safety = med AND
maint = vhigh AND
doors = 5more: acc (4.0)

safety = med AND
maint = vhigh AND
persons = 4: unacc (2.0)

safety = med AND
maint = high AND
lug_boot = small: unacc (8.0)

safety = med AND
buying = high AND
doors = 2: unacc (6.0)

safety = med AND
lug_boot = small AND
doors = 3: acc (8.0)

safety = med AND
maint = high AND
lug_boot = big: acc (8.0)

lug_boot = big AND
safety = high AND
maint = med: vgood (16.0)

maint = vhigh AND
lug_boot = med: acc (14.0)

maint = vhigh AND
doors = 4: acc (8.0)

safety = med AND
maint = med AND
buying = med: acc (22.0/1.0)

maint = vhigh AND
doors = 5more: acc (8.0)

lug_boot = big AND
safety = high AND
maint = low: vgood (16.0)
```

maint = vhigh AND
persons = 4: acc (4.0)

maint = vhigh AND
lug_boot = small: unacc (2.0)

maint = high AND
buying = med: acc (32.0/4.0)

lug_boot = big AND
maint = high: vgood (8.0)

lug_boot = big AND
maint = low: good (8.0)

lug_boot = small AND
safety = med AND
doors = 4: acc (6.0)

lug_boot = small AND
persons = 4 AND
safety = high AND
maint = low: good (9.0/1.0)

lug_boot = small AND
persons = 4 AND
maint = high: acc (5.0)

lug_boot = small AND
doors = 5more AND
safety = med: acc (6.0)

lug_boot = small AND
doors = 3 AND
maint = med AND
buying = med: acc (2.0)

lug_boot = small AND
doors = 3: good (5.0/1.0)

lug_boot = small AND
doors = 4 AND
maint = med AND
buying = med: acc (2.0)

lug_boot = small AND
doors = 4: good (5.0/1.0)

lug_boot = small AND
persons = 4 AND
buying = med: acc (3.0)

lug_boot = small AND
doors = 2 AND
persons = more: unacc (11.0)

safety = med AND
doors = 2: acc (8.0)

```
safety = med AND
buying = med: good (6.0/1.0)

safety = med AND
buying = low AND
doors = 4: good (4.0)

doors = 4: vgood (10.0)

buying = high AND
doors = 2: acc (7.0)

safety = med AND
buying = low AND
doors = 5more: good (4.0)

safety = med AND
persons = 4 AND
buying = high: unacc (3.0)

lug_boot = med AND
doors = 5more: vgood (10.0)

maint = high: acc (6.0/1.0)

lug_boot = small: good (6.0/1.0)

maint = low AND
safety = high AND
doors = 2: good (4.0)

persons = 4 AND
safety = high AND
buying = med: acc (3.0/1.0)

buying = high: acc (2.0)

lug_boot = med AND
buying = med: vgood (3.0/1.0)

lug_boot = med AND
safety = high AND
persons = 4: good (3.0)

persons = more AND
lug_boot = med AND
safety = high: vgood (3.0/1.0)

doors = 2: acc (2.0)

persons = 4: acc (2.0)

: good (2.0)

Number of Rules  :         68


Time taken to build model: 0.03 seconds
```

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1655               95.7755 %
Incorrectly Classified Instances        73                4.2245 %
Kappa statistic                          0.9091
Mean absolute error                      0.0241
Root mean squared error                  0.1276
Relative absolute error                 10.5343 %
Root relative squared error             37.7421 %
Total Number of Instances             1728

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,975    0,012    0,995      0,975    0,985      0,952    0,992     0,996     unacc
               0,938    0,031    0,896      0,938    0,916      0,892    0,987     0,957     acc
               0,739    0,012    0,718      0,739    0,729      0,717    0,980     0,836     good
               0,985    0,003    0,928      0,985    0,955      0,954    0,999     0,961     vgood
Weighted Avg.  0,958    0,016    0,959      0,958    0,958      0,929    0,990     0,979

=== Confusion Matrix ===

   a    b    c    d   <-- classified as
 1180   26    4    0 |    a = unacc
   6  360   16    2 |    b = acc
   0   15   51    3 |    c = good
   0    1    0   64 |    d = vgood
```

Apêndice C

```
J48
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     car
Instances:    1728
Attributes:   7
              buying
              maint
              doors
              persons
              lug_boot
              safety
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

safety = low: unacc (576.0)
safety = med
|   persons = 2: unacc (192.0)
|   persons = 4
```

```
|   |   buying = vhigh
|   |   |   maint = vhigh: unacc (12.0)
|   |   |   maint = high: unacc (12.0)
|   |   |   maint = med
|   |   |   |   lug_boot = small: unacc (4.0)
|   |   |   |   lug_boot = med: unacc (4.0/2.0)
|   |   |   |   lug_boot = big: acc (4.0)
|   |   |   maint = low
|   |   |   |   lug_boot = small: unacc (4.0)
|   |   |   |   lug_boot = med: unacc (4.0/2.0)
|   |   |   |   lug_boot = big: acc (4.0)
|   |   buying = high
|   |   |   lug_boot = small: unacc (16.0)
|   |   |   lug_boot = med
|   |   |   |   doors = 2: unacc (4.0)
|   |   |   |   doors = 3: unacc (4.0)
|   |   |   |   doors = 4: acc (4.0/1.0)
|   |   |   |   doors = 5more: acc (4.0/1.0)
|   |   |   lug_boot = big
|   |   |   |   maint = vhigh: unacc (4.0)
|   |   |   |   maint = high: acc (4.0)
|   |   |   |   maint = med: acc (4.0)
|   |   |   |   maint = low: acc (4.0)
|   |   buying = med
|   |   |   maint = vhigh
|   |   |   |   lug_boot = small: unacc (4.0)
|   |   |   |   lug_boot = med: unacc (4.0/2.0)
|   |   |   |   lug_boot = big: acc (4.0)
|   |   |   maint = high
|   |   |   |   lug_boot = small: unacc (4.0)
|   |   |   |   lug_boot = med: unacc (4.0/2.0)
|   |   |   |   lug_boot = big: acc (4.0)
|   |   |   maint = med: acc (12.0)
|   |   |   maint = low
|   |   |   |   lug_boot = small: acc (4.0)
|   |   |   |   lug_boot = med: acc (4.0/2.0)
|   |   |   |   lug_boot = big: good (4.0)
|   |   buying = low
|   |   |   maint = vhigh
|   |   |   |   lug_boot = small: unacc (4.0)
|   |   |   |   lug_boot = med: unacc (4.0/2.0)
|   |   |   |   lug_boot = big: acc (4.0)
|   |   |   maint = high: acc (12.0)
|   |   |   maint = med
|   |   |   |   lug_boot = small: acc (4.0)
|   |   |   |   lug_boot = med: acc (4.0/2.0)
|   |   |   |   lug_boot = big: good (4.0)
|   |   |   maint = low
|   |   |   |   lug_boot = small: acc (4.0)
|   |   |   |   lug_boot = med: acc (4.0/2.0)
|   |   |   |   lug_boot = big: good (4.0)
|   persons = more
|   |   lug_boot = small
|   |   |   buying = vhigh: unacc (16.0)
|   |   |   buying = high: unacc (16.0)
|   |   |   buying = med
|   |   |   |   maint = vhigh: unacc (4.0)
|   |   |   |   maint = high: unacc (4.0)
|   |   |   |   maint = med: acc (4.0/1.0)
```

```
|   |   |   |   maint = low: acc (4.0/1.0)
|   |   |   buying = low
|   |   |   |   maint = vhigh: unacc (4.0)
|   |   |   |   maint = high: acc (4.0/1.0)
|   |   |   |   maint = med: acc (4.0/1.0)
|   |   |   |   maint = low: acc (4.0/1.0)
|   |   lug_boot = med
|   |   |   buying = vhigh
|   |   |   |   maint = vhigh: unacc (4.0)
|   |   |   |   maint = high: unacc (4.0)
|   |   |   |   maint = med: acc (4.0/1.0)
|   |   |   |   maint = low: acc (4.0/1.0)
|   |   |   buying = high
|   |   |   |   maint = vhigh: unacc (4.0)
|   |   |   |   maint = high: acc (4.0/1.0)
|   |   |   |   maint = med: acc (4.0/1.0)
|   |   |   |   maint = low: acc (4.0/1.0)
|   |   |   buying = med: acc (16.0/5.0)
|   |   |   buying = low
|   |   |   |   maint = vhigh: acc (4.0/1.0)
|   |   |   |   maint = high: acc (4.0)
|   |   |   |   maint = med: good (4.0/1.0)
|   |   |   |   maint = low: good (4.0/1.0)
|   |   lug_boot = big
|   |   |   buying = vhigh
|   |   |   |   maint = vhigh: unacc (4.0)
|   |   |   |   maint = high: unacc (4.0)
|   |   |   |   maint = med: acc (4.0)
|   |   |   |   maint = low: acc (4.0)
|   |   |   buying = high
|   |   |   |   maint = vhigh: unacc (4.0)
|   |   |   |   maint = high: acc (4.0)
|   |   |   |   maint = med: acc (4.0)
|   |   |   |   maint = low: acc (4.0)
|   |   |   buying = med
|   |   |   |   maint = vhigh: acc (4.0)
|   |   |   |   maint = high: acc (4.0)
|   |   |   |   maint = med: acc (4.0)
|   |   |   |   maint = low: good (4.0)
|   |   |   buying = low
|   |   |   |   maint = vhigh: acc (4.0)
|   |   |   |   maint = high: acc (4.0)
|   |   |   |   maint = med: good (4.0)
|   |   |   |   maint = low: good (4.0)
safety = high
|   persons = 2: unacc (192.0)
|   persons = 4
|   |   buying = vhigh
|   |   |   maint = vhigh: unacc (12.0)
|   |   |   maint = high: unacc (12.0)
|   |   |   maint = med: acc (12.0)
|   |   |   maint = low: acc (12.0)
|   |   buying = high
|   |   |   maint = vhigh: unacc (12.0)
|   |   |   maint = high: acc (12.0)
|   |   |   maint = med: acc (12.0)
|   |   |   maint = low: acc (12.0)
|   |   buying = med
|   |   |   maint = vhigh: acc (12.0)
```

```
|   |   |   maint = high: acc (12.0)
|   |   |   maint = med
|   |   |   |   lug_boot = small: acc (4.0)
|   |   |   |   lug_boot = med: acc (4.0/2.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   |   maint = low
|   |   |   |   lug_boot = small: good (4.0)
|   |   |   |   lug_boot = med: good (4.0/2.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   buying = low
|   |   |   maint = vhigh: acc (12.0)
|   |   |   maint = high
|   |   |   |   lug_boot = small: acc (4.0)
|   |   |   |   lug_boot = med: acc (4.0/2.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   |   maint = med
|   |   |   |   lug_boot = small: good (4.0)
|   |   |   |   lug_boot = med: good (4.0/2.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   |   maint = low
|   |   |   |   lug_boot = small: good (4.0)
|   |   |   |   lug_boot = med: good (4.0/2.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   persons = more
|   |   buying = vhigh
|   |   |   maint = vhigh: unacc (12.0)
|   |   |   maint = high: unacc (12.0)
|   |   |   maint = med: acc (12.0/1.0)
|   |   |   maint = low: acc (12.0/1.0)
|   |   buying = high
|   |   |   maint = vhigh: unacc (12.0)
|   |   |   maint = high: acc (12.0/1.0)
|   |   |   maint = med: acc (12.0/1.0)
|   |   |   maint = low: acc (12.0/1.0)
|   |   buying = med
|   |   |   maint = vhigh: acc (12.0/1.0)
|   |   |   maint = high: acc (12.0/1.0)
|   |   |   maint = med
|   |   |   |   lug_boot = small: acc (4.0/1.0)
|   |   |   |   lug_boot = med: vgood (4.0/1.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   |   maint = low
|   |   |   |   lug_boot = small: good (4.0/1.0)
|   |   |   |   lug_boot = med: vgood (4.0/1.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   buying = low
|   |   |   maint = vhigh: acc (12.0/1.0)
|   |   |   maint = high
|   |   |   |   lug_boot = small: acc (4.0/1.0)
|   |   |   |   lug_boot = med: vgood (4.0/1.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   |   maint = med
|   |   |   |   lug_boot = small: good (4.0/1.0)
|   |   |   |   lug_boot = med: vgood (4.0/1.0)
|   |   |   |   lug_boot = big: vgood (4.0)
|   |   |   maint = low
|   |   |   |   lug_boot = small: good (4.0/1.0)
|   |   |   |   lug_boot = med: vgood (4.0/1.0)
|   |   |   |   lug_boot = big: vgood (4.0)
```

```
Number of Leaves  :      131

Size of the tree :      182


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1596               92.3611 %
Incorrectly Classified Instances       132               7.6389 %
Kappa statistic                          0.8343
Mean absolute error                      0.0421
Root mean squared error                  0.1718
Relative absolute error                 18.3833 %
Root relative squared error             50.8176 %
Total Number of Instances             1728

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,962    0,064    0,972      0,962    0,967      0,892    0,983     0,992     unacc
               0,867    0,047    0,841      0,867    0,854      0,811    0,962     0,859     acc
               0,609    0,011    0,689      0,609    0,646      0,634    0,918     0,593     good
               0,877    0,010    0,770      0,877    0,820      0,814    0,995     0,808     vgood
Weighted Avg.    0,924    0,056    0,924      0,924    0,924      0,861    0,976     0,940

=== Confusion Matrix ===

   a    b    c    d   <-- classified as
 1164   43    3    0 |    a = unacc
  33  333   11    7 |    b = acc
   0   17   42   10 |    c = good
   0    3    5   57 |    d = vgood
```

Apêndice D

```
LMT
=== Run information ===

Scheme:       weka.classifiers.trees.LMT -I -1 -M 15 -W 0.0
Relation:     car
Instances:    1728
Attributes:   7
              buying
              maint
              doors
              persons
              lug_boot
              safety
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===
```

```
Logistic model tree
------------------

safety = low: LM_1:22/156 (576)
safety = med
|   persons = 2: LM_2:20/288 (192)
|   persons = 4
|   |   buying = vhigh: LM_3:134/536 (48)
|   |   buying = high: LM_4:134/536 (48)
|   |   buying = med: LM_5:134/536 (48)
|   |   buying = low: LM_6:134/536 (48)
|   persons = more
|   |   lug_boot = small: LM_7:134/536 (64)
|   |   lug_boot = med: LM_8:134/536 (64)
|   |   lug_boot = big: LM_9:134/536 (64)
safety = high
|   persons = 2: LM_10:20/288 (192)
|   persons = 4: LM_11:134/402 (192)
|   persons = more
|   |   buying = vhigh: LM_12:134/536 (48)
|   |   buying = high: LM_13:134/536 (48)
|   |   buying = med: LM_14:134/536 (48)
|   |   buying = low: LM_15:134/536 (48)

Number of Leaves  :      15

Size of the Tree :       21

LM_1:
Class unacc :
15.56 +
[buying=vhigh] * 2.14 +
[buying=high] * 1.23 +
[buying=med] * -0.62 +
[buying=low] * -3.21 +
[maint=vhigh] * 2.11 +
[maint=high] * 1.03 +
[maint=med] * -0.5 +
[maint=low] * -1.85 +
[doors=2] * 1.85 +
[doors=3] * 0.44 +
[persons=2] * 24.44 +
[lug_boot=small] * 1.8  +
[lug_boot=big] * -1.78 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
-15.6 +
[buying=vhigh] * -1.06 +
[buying=high] * -0.16 +
[buying=med] * 1.52 +
[buying=low] * -0.12 +
[maint=vhigh] * -1.79 +
[maint=high] * -0.39 +
[maint=med] * 1.1  +
[maint=low] * -0.31 +
[doors=2] * -0.3 +
[doors=3] * -0.02 +
```

[persons=2] * -3.71 +
[persons=4] * 0.32 +
[lug_boot=small] * -0.63 +
[lug_boot=med] * 0.17 +
[lug_boot=big] * -0.29 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-15.32 +
[buying=vhigh] * -12.91 +
[buying=high] * -11.37 +
[buying=low] * 1.26 +
[maint=vhigh] * -13.03 +
[maint=high] * -10.56 +
[maint=low] * 1.03 +
[doors=2] * -1.82 +
[doors=3] * -0.29 +
[doors=4] * 0.11 +
[doors=5more] * 0.13 +
[persons=2] * -0.98 +
[persons=4] * 0.33 +
[lug_boot=small] * -3.89 +
[lug_boot=big] * 0.84 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-23.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.17 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_2:
Class unacc :
13.68 +
[buying=vhigh] * 2.89 +
[buying=high] * 1.7 +
[buying=med] * -0.91 +
[buying=low] * -5.29 +
[maint=vhigh] * 2.93 +
[maint=high] * 1.21 +
[maint=med] * -0.59 +
[maint=low] * -2.61 +
[doors=2] * 2.53 +
[doors=3] * 0.48 +

[doors=4] * -1.14 +
[doors=5more] * -1.01 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 4.05 +
[lug_boot=med] * 0.59 +
[lug_boot=big] * -3.55 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
-13.68 +
[buying=vhigh] * -2.2 +
[buying=high] * -0.7 +
[buying=med] * 2.44 +
[buying=low] * 0.5 +
[maint=vhigh] * -3.51 +
[maint=high] * -0.7 +
[maint=med] * 1.74 +
[maint=low] * -0.29 +
[doors=2] * -0.44 +
[doors=3] * -0.02 +
[doors=4] * 0.15 +
[doors=5more] * 0.27 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -2.5 +
[lug_boot=med] * 0.17 +
[lug_boot=big] * -0.21 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-14.73 +
[buying=vhigh] * -18.23 +
[buying=high] * -16.68 +
[buying=low] * 2.35 +
[maint=vhigh] * -19.12 +
[maint=high] * -15.89 +
[maint=low] * 2.27 +
[doors=2] * -2.82 +
[doors=3] * -0.33 +
[doors=4] * 1.85 +
[doors=5more] * 1.97 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -17.53 +
[lug_boot=big] * 2.81 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-122.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +

[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_3:
Class unacc :
-9.21 +
[buying=vhigh] * 3.51 +
[buying=high] * 1.8 +
[buying=med] * -1.28 +
[buying=low] * -7.71 +
[maint=vhigh] * 28.12 +
[maint=high] * 29.25 +
[maint=med] * -0.92 +
[maint=low] * -3.27 +
[doors=2] * 14.88 +
[doors=3] * 15.16 +
[doors=4] * -1.38 +
[doors=5more] * -1.38 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 17.97 +
[lug_boot=med] * 1.17 +
[lug_boot=big] * -16.3 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
8.42 +
[buying=vhigh] * -2.9 +
[buying=high] * -1.07 +
[buying=med] * 3.16 +
[buying=low] * 1.26 +
[maint=vhigh] * -28.92 +
[maint=high] * -28.77 +
[maint=med] * 2.22 +
[maint=low] * -0.15 +
[doors=2] * -12.69 +
[doors=3] * -12.91 +
[doors=4] * 0.32 +
[doors=5more] * 0.35 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -15.6 +
[lug_boot=med] * -0.18 +
[lug_boot=big] * 11.82 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-101.48 +
[buying=vhigh] * -20.5 +
[buying=high] * -18.19 +
[buying=low] * 3.62 +
[maint=vhigh] * -21.41 +
[maint=high] * -17.4 +
[maint=low] * 3.16 +
[doors=2] * -3.28 +
[doors=3] * -3.26 +
[doors=4] * 2.96 +
[doors=5more] * 2.98 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -17.53 +
[lug_boot=big] * 5.83 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-308.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_4:
Class unacc :
8.59 +
[buying=vhigh] * 3.51 +
[buying=high] * 1.8 +
[buying=med] * -1.28 +
[buying=low] * -7.71 +
[maint=vhigh] * 42.1 +
[maint=high] * -1.33 +
[maint=med] * -0.92 +
[maint=low] * -3.27 +
[doors=2] * 2.58 +
[doors=3] * 2.62 +
[doors=4] * -21.11 +
[doors=5more] * -20.38 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 24.41 +
[lug_boot=med] * 1.17 +

[lug_boot=big] * -23.58 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
-9.38 +
[buying=vhigh] * -2.9 +
[buying=high] * -1.07 +
[buying=med] * 3.16 +
[buying=low] * 1.26 +
[maint=vhigh] * -42.89 +
[maint=high] * 1.82 +
[maint=med] * 2.22 +
[maint=low] * -0.15 +
[doors=2] * -0.4 +
[doors=3] * -0.37 +
[doors=4] * 20.04 +
[doors=5more] * 19.36 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -22.04 +
[lug_boot=med] * -0.18 +
[lug_boot=big] * 19.1 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-101.48 +
[buying=vhigh] * -20.5 +
[buying=high] * -18.19 +
[buying=low] * 3.62 +
[maint=vhigh] * -21.41 +
[maint=high] * -17.4 +
[maint=low] * 3.16 +
[doors=2] * -3.28 +
[doors=3] * -3.26 +
[doors=4] * 2.96 +
[doors=5more] * 2.98 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -17.53 +
[lug_boot=big] * 5.83 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-308.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +

[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_5:
Class unacc :
-4.42 +
[buying=vhigh] * 3.51 +
[buying=high] * 1.8 +
[buying=med] * -1.28 +
[buying=low] * -7.71 +
[maint=vhigh] * 3.98 +
[maint=high] * 5.06 +
[maint=med] * -0.92 +
[maint=low] * -38.46 +
[doors=2] * 12.49 +
[doors=3] * 12.05 +
[doors=4] * -5.51 +
[doors=5more] * -5.93 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 12.79 +
[lug_boot=med] * 1.17 +
[lug_boot=big] * -25.97 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
0.92 +
[buying=vhigh] * -2.9 +
[buying=high] * -1.07 +
[buying=med] * 3.16 +
[buying=low] * 1.26 +
[maint=vhigh] * -4.77 +
[maint=high] * -3.59 +
[maint=med] * 39.24 +
[maint=low] * -0.37 +
[doors=2] * -0.96 +
[doors=3] * -1.23 +
[doors=4] * 4.12 +
[doors=5more] * 3.7 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -10.16 +
[lug_boot=med] * 1.59 +
[lug_boot=big] * -0.84 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-33.58 +
[buying=vhigh] * -20.5 +
[buying=high] * -18.19 +
[buying=low] * 3.62 +
[maint=vhigh] * -21.41 +

[maint=high] * -17.4 +
[maint=low] * 31.43 +
[doors=2] * -4.39 +
[doors=3] * -4.59 +
[doors=4] * 22.22 +
[doors=5more] * 21.63 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -34.52 +
[lug_boot=big] * 20.38 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-308.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_6:
Class unacc :
-34.12 +
[buying=vhigh] * 3.51 +
[buying=high] * 1.8 +
[buying=med] * -1.28 +
[buying=low] * -7.71 +
[maint=vhigh] * 36.91 +
[maint=high] * 1.64 +
[maint=med] * -0.92 +
[maint=low] * -3.27 +
[doors=2] * 12.87 +
[doors=3] * 12.69 +
[doors=4] * -7.6 +
[doors=5more] * -7.33 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 14.25 +
[lug_boot=med] * 1.17 +
[lug_boot=big] * -26.28 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
-0.92 +
[buying=vhigh] * -2.9 +

[buying=high] * -1.07 +
[buying=med] * 3.16 +
[buying=low] * 1.26 +
[maint=vhigh] * -4.77 +
[maint=high] * 30.31 +
[maint=med] * 2.42 +
[maint=low] * 2.18 +
[doors=2] * -0.09 +
[doors=3] * -0.3 +
[doors=4] * 0.93 +
[doors=5more] * 0.85 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -6.42 +
[lug_boot=med] * 2.72 +
[lug_boot=big] * -2.31 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-4.99 +
[buying=vhigh] * -20.5 +
[buying=high] * -18.19 +
[buying=low] * 3.62 +
[maint=vhigh] * -50.48 +
[maint=high] * -17.4 +
[maint=med] * 3.26 +
[maint=low] * 3.16 +
[doors=2] * -7.82 +
[doors=3] * -8.08 +
[doors=4] * 15.63 +
[doors=5more] * 15.52 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -33.23 +
[lug_boot=big] * 18.58 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-308.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_7:
Class unacc :
-8.91 +
[buying=vhigh] * 15.3 +
[buying=high] * 14.82 +
[buying=med] * -1.27 +
[buying=low] * -21.38 +
[maint=vhigh] * 30.77 +
[maint=high] * 14.29 +
[maint=med] * -0.82 +
[maint=low] * -3.56 +
[doors=2] * 32.99 +
[doors=3] * -1.19 +
[doors=4] * -1.14 +
[doors=5more] * -1.01 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 5.8 +
[lug_boot=med] * -0.07 +
[lug_boot=big] * -5.12 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
8.36 +
[buying=vhigh] * -14.53 +
[buying=high] * -14.04 +
[buying=med] * 3 +
[buying=low] * 14.75 +
[maint=vhigh] * -31.31 +
[maint=high] * -13.48 +
[maint=med] * 2.67 +
[maint=low] * -0.11 +
[doors=2] * -29.72 +
[doors=3] * 0.15 +
[doors=4] * 0.15 +
[doors=5more] * 0.27 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -5.23 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 0.44 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-100.78 +
[buying=vhigh] * -21.28 +
[buying=high] * -18.98 +
[buying=low] * 4.27 +
[maint=vhigh] * -22.16 +
[maint=high] * -18.17 +
[maint=low] * 3.92 +
[doors=2] * -5.59 +
[doors=3] * 1.83 +
[doors=4] * 1.85 +

[doors=5more] * 1.97 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -21.36 +
[lug_boot=med] * 1.03 +
[lug_boot=big] * 5.15 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-308.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_8:
Class unacc :
-12.12 +
[buying=vhigh] * 11.73 +
[buying=high] * 3.59 +
[buying=med] * -5.42 +
[buying=low] * -25.55 +
[maint=vhigh] * 11.43 +
[maint=high] * 4.38 +
[maint=med] * -5.39 +
[maint=low] * -10.59 +
[doors=2] * 23.36 +
[doors=3] * -1.19 +
[doors=4] * -1.14 +
[doors=5more] * -1.01 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 5.8 +
[lug_boot=med] * -0.07 +
[lug_boot=big] * -5.12 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
6.92 +
[buying=vhigh] * -11.54 +
[buying=high] * -2.5 +
[buying=med] * 6.87 +
[buying=low] * 2.91 +
[maint=vhigh] * -13.12 +

[maint=high] * -2.5 +
[maint=med] * 5.46 +
[maint=low] * 0.43 +
[doors=2] * -10.41 +
[doors=3] * 0.15 +
[doors=4] * 0.15 +
[doors=5more] * 0.27 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -5.23 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 0.44 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
6.79 +
[buying=vhigh] * -37.65 +
[buying=high] * -28.19 +
[buying=low] * 13.92 +
[maint=vhigh] * -37.56 +
[maint=high] * -26.75 +
[maint=low] * 13.16 +
[doors=2] * -44.88 +
[doors=3] * 1.83 +
[doors=4] * 1.85 +
[doors=5more] * 1.97 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -21.36 +
[lug_boot=med] * 1.03 +
[lug_boot=big] * 5.15 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-308.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_9:
Class unacc :
-7.99 +

[buying=vhigh] * 18.48 +
[buying=high] * 2.91 +
[buying=med] * -16.61 +
[buying=low] * -26.17 +
[maint=vhigh] * 17.51 +
[maint=high] * 3.26 +
[maint=med] * -15.15 +
[maint=low] * -22.62 +
[doors=2] * -0.79 +
[doors=3] * -1.19 +
[doors=4] * -1.14 +
[doors=5more] * -1.01 +
[persons=2] * 24.44 +
[persons=more] * -0.5 +
[lug_boot=small] * 5.8 +
[lug_boot=med] * -0.07 +
[lug_boot=big] * -5.12 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
6.64 +
[buying=vhigh] * -11.05 +
[buying=high] * -1.39 +
[buying=med] * 5.52 +
[buying=low] * -3.93 +
[maint=vhigh] * -12.9 +
[maint=high] * -1.64 +
[maint=med] * 6.56 +
[maint=low] * -1.24 +
[doors=2] * 0.05 +
[doors=3] * 0.15 +
[doors=4] * 0.15 +
[doors=5more] * 0.27 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[persons=more] * 0.13 +
[lug_boot=small] * -5.23 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 0.44 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-1.2 +
[buying=vhigh] * -43.02 +
[buying=high] * -34.07 +
[buying=low] * 15.79 +
[maint=vhigh] * -45.83 +
[maint=high] * -34.32 +
[maint=low] * 17.48 +
[doors=2] * 1.1 +
[doors=3] * 1.83 +
[doors=4] * 1.85 +
[doors=5more] * 1.97 +
[persons=2] * -7.06 +
[persons=4] * 0.33 +
[persons=more] * 1.17 +
[lug_boot=small] * -21.36 +

[lug_boot=med] * 1.03 +
[lug_boot=big] * 5.15 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-308.16 +
[buying=vhigh] * -14.81 +
[buying=high] * -13.83 +
[buying=low] * 2.16 +
[maint=vhigh] * -16.91 +
[maint=high] * -7.48 +
[maint=med] * -0 +
[maint=low] * 0.61 +
[doors=2] * -3.73 +
[doors=3] * -1.21 +
[doors=4] * 0.5 +
[doors=5more] * 0.56 +
[persons=2] * -1.94 +
[persons=more] * 0.3 +
[lug_boot=small] * -10.54 +
[lug_boot=big] * 2.76 +
[safety=high] * 13.39

LM_10:
Class unacc :
14.05 +
[buying=vhigh] * 2.66 +
[buying=high] * 1.23 +
[buying=med] * -0.62 +
[buying=low] * -5.58 +
[maint=vhigh] * 2.51 +
[maint=high] * 1.08 +
[maint=med] * -0.5 +
[maint=low] * -4.38 +
[doors=2] * 1.97 +
[doors=3] * 0 +
[doors=5more] * -0.03 +
[persons=2] * 36.58 +
[persons=4] * -0.88 +
[lug_boot=small] * 1.91 +
[lug_boot=med] * 0.54 +
[lug_boot=big] * -0.32 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
-13.39 +
[buying=vhigh] * -1.58 +
[buying=high] * -0.12 +
[buying=med] * 2.15 +
[buying=low] * -2.85 +
[maint=vhigh] * -1.84 +
[maint=high] * -0.37 +
[maint=med] * 2.14 +
[maint=low] * -1.76 +
[doors=2] * -0.21 +
[doors=3] * 0.02 +

[doors=5more] * -0.02 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[lug_boot=small] * -0.63 +
[lug_boot=med] * -0.27 +
[lug_boot=big] * -1.14 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-16.32 +
[buying=vhigh] * -18.18 +
[buying=high] * -16.66 +
[buying=low] * 2.91 +
[maint=vhigh] * -19.85 +
[maint=high] * -17.4 +
[maint=low] * 2.71 +
[doors=2] * -2.41 +
[doors=3] * 0.19 +
[doors=4] * -0.29 +
[doors=5more] * -0.3 +
[persons=2] * -0.98 +
[persons=4] * 2.21 +
[lug_boot=small] * -2.62 +
[lug_boot=med] * 1.77 +
[lug_boot=big] * -6.1 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-21.15 +
[buying=vhigh] * -23.28 +
[buying=high] * -22.26 +
[buying=low] * 3.31 +
[maint=vhigh] * -26.2 +
[maint=high] * -11.3 +
[maint=low] * 1.64 +
[doors=2] * -5.94 +
[doors=3] * -1.42 +
[doors=4] * 1.36 +
[doors=5more] * 1.29 +
[persons=2] * -1.94 +
[persons=more] * 0.49 +
[lug_boot=small] * -15.15 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 4.61 +
[safety=high] * 13.39

LM_11:
Class unacc :
-4.75 +
[buying=vhigh] * 16.27 +
[buying=high] * 2.59 +
[buying=med] * -11.4 +
[buying=low] * -26.64 +
[maint=vhigh] * 13.15 +
[maint=high] * 2.45 +
[maint=med] * -18.03 +
[maint=low] * -26.87 +

[doors=2] * 0.5 +
[doors=3] * 0   +
[doors=5more] * -0.03 +
[persons=2] * 36.58 +
[persons=4] * -0.88 +
[lug_boot=small] * 1.3 +
[lug_boot=med] * 1.12 +
[lug_boot=big] * -0.86 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
3.49 +
[buying=vhigh] * -5.25 +
[buying=high] * -0.68 +
[buying=med] * 4   +
[buying=low] * -11.2 +
[maint=vhigh] * -3.38 +
[maint=high] * 3.94 +
[maint=med] * 2.38 +
[maint=low] * -6.64 +
[doors=2] * 0.77 +
[doors=3] * 0.73 +
[doors=5more] * -0.29 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[lug_boot=small] * 0.24 +
[lug_boot=med] * 0.13 +
[lug_boot=big] * -2.33 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-2.57 +
[buying=vhigh] * -27.48 +
[buying=high] * -22.72 +
[buying=low] * 9.35 +
[maint=vhigh] * -34.85 +
[maint=high] * -26.84 +
[maint=low] * 5.16 +
[doors=2] * 0.81 +
[doors=3] * 0.59 +
[doors=4] * -4.12 +
[doors=5more] * -4.29 +
[persons=2] * -0.98 +
[persons=4] * 2.21 +
[lug_boot=small] * 2.3 +
[lug_boot=med] * 1.77 +
[lug_boot=big] * -9.14 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-1.76 +
[buying=vhigh] * -39.16 +
[buying=high] * -34.78 +
[buying=low] * 4.89 +
[maint=vhigh] * -45.14 +
[maint=high] * -16.37 +

[maint=med] * 1.74 +
[maint=low] * 1.64 +
[doors=2] * -7.7 +
[doors=3] * -7.98 +
[doors=4] * 1.64 +
[doors=5more] * 1.56 +
[persons=2] * -1.94 +
[persons=more] * 0.49 +
[lug_boot=small] * -15.15 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 8.21 +
[safety=high] * 13.39

LM_12:
Class unacc :
-27.9 +
[buying=vhigh] * 2.29 +
[buying=high] * 1.21 +
[buying=med] * -0.47 +
[buying=low] * -5.69 +
[maint=vhigh] * 37.99 +
[maint=high] * 37.88 +
[maint=med] * -0.68 +
[maint=low] * -4.09 +
[doors=2] * 20.52 +
[doors=3] * -0.01 +
[doors=5more] * -0.03 +
[persons=2] * 36.58 +
[persons=4] * -0.88 +
[lug_boot=small] * 19.58 +
[lug_boot=med] * 0.56 +
[lug_boot=big] * -0.41 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
28.04 +
[buying=vhigh] * -1.37 +
[buying=high] * -0.12 +
[buying=med] * 1.63 +
[buying=low] * -3.65 +
[maint=vhigh] * -37.5 +
[maint=high] * -37.01 +
[maint=med] * 1.32 +
[maint=low] * -2.11 +
[doors=2] * -16.82 +
[doors=3] * -0.01 +
[doors=5more] * -0.02 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[lug_boot=small] * -17.79 +
[lug_boot=med] * -0.27 +
[lug_boot=big] * -1.24 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-104.46 +

[buying=vhigh] * -21.16 +
[buying=high] * -19.65 +
[buying=med] * 0.71 +
[buying=low] * 3.4 +
[maint=vhigh] * -23.6 +
[maint=high] * -18.14 +
[maint=med] * 1.16 +
[maint=low] * 2.8 +
[doors=2] * -5.95 +
[doors=3] * 0.62 +
[doors=4] * 0.68 +
[doors=5more] * 0.6 +
[persons=2] * -0.98 +
[persons=4] * 2.21 +
[lug_boot=small] * -2.62 +
[lug_boot=med] * 6.92 +
[lug_boot=big] * -6.1 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-105.06 +
[buying=vhigh] * -27.83 +
[buying=high] * -26.04 +
[buying=low] * 5.28 +
[maint=vhigh] * -32.99 +
[maint=high] * -14.77 +
[maint=med] * 0.49 +
[maint=low] * 1.76 +
[doors=2] * -9.76 +
[doors=3] * 1.29 +
[doors=4] * 1.36 +
[doors=5more] * 1.29 +
[persons=2] * -1.94 +
[persons=more] * 0.49 +
[lug_boot=small] * -21.23 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 5.75 +
[safety=high] * 13.39

LM_13:
Class unacc :
-50.73 +
[buying=vhigh] * 2.29 +
[buying=high] * 1.21 +
[buying=med] * -0.47 +
[buying=low] * -5.69 +
[maint=vhigh] * 69.4 +
[maint=high] * -0.73 +
[maint=med] * -0.68 +
[maint=low] * -4.09 +
[doors=2] * 36.09 +
[doors=3] * -0.01 +
[doors=5more] * -0.03 +
[persons=2] * 36.58 +
[persons=4] * -0.88 +
[lug_boot=small] * 35.87 +
[lug_boot=med] * 0.56 +
[lug_boot=big] * -0.41 +

```
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
50.85 +
[buying=vhigh] * -1.37 +
[buying=high] * -0.12 +
[buying=med] * 1.63 +
[buying=low] * -3.65 +
[maint=vhigh] * -68.92 +
[maint=high] * 1.6  +
[maint=med] * 1.32 +
[maint=low] * -2.11 +
[doors=2] * -32.39 +
[doors=3] * -0.01 +
[doors=5more] * -0.02 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[lug_boot=small] * -34.04 +
[lug_boot=med] * -0.27 +
[lug_boot=big] * -1.24 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-104.46 +
[buying=vhigh] * -21.16 +
[buying=high] * -19.65 +
[buying=med] * 0.71 +
[buying=low] * 3.4  +
[maint=vhigh] * -23.6 +
[maint=high] * -18.14 +
[maint=med] * 1.16 +
[maint=low] * 2.8  +
[doors=2] * -5.95 +
[doors=3] * 0.62 +
[doors=4] * 0.68 +
[doors=5more] * 0.6  +
[persons=2] * -0.98 +
[persons=4] * 2.21 +
[lug_boot=small] * -2.62 +
[lug_boot=med] * 6.92 +
[lug_boot=big] * -6.1 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
-105.06 +
[buying=vhigh] * -27.83 +
[buying=high] * -26.04 +
[buying=low] * 5.28 +
[maint=vhigh] * -32.99 +
[maint=high] * -14.77 +
[maint=med] * 0.49 +
[maint=low] * 1.76 +
[doors=2] * -9.76 +
[doors=3] * 1.29 +
[doors=4] * 1.36 +
[doors=5more] * 1.29 +
```

[persons=2] * -1.94 +
[persons=more] * 0.49 +
[lug_boot=small] * -21.23 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 5.75 +
[safety=high] * 13.39

LM_14:
Class unacc :
-91.02 +
[buying=vhigh] * 2.29 +
[buying=high] * 1.21 +
[buying=med] * -0.47 +
[buying=low] * -5.69 +
[maint=vhigh] * -0.26 +
[maint=high] * 1.14 +
[maint=med] * -7.05 +
[maint=low] * -2.93 +
[doors=2] * 79.68 +
[doors=3] * -0.01 +
[doors=5more] * -0.03 +
[persons=2] * 36.58 +
[persons=4] * -0.88 +
[lug_boot=small] * 75.81 +
[lug_boot=med] * 0.56 +
[lug_boot=big] * -0.41 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
2.5 +
[buying=vhigh] * -1.37 +
[buying=high] * -0.12 +
[buying=med] * 1.63 +
[buying=low] * -3.65 +
[maint=vhigh] * 21.99 +
[maint=high] * 23.57 +
[maint=med] * -1.33 +
[maint=low] * -15.08 +
[doors=2] * 6.43 +
[doors=3] * -0.01 +
[doors=5more] * -0.02 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[lug_boot=small] * 0.69 +
[lug_boot=med] * 3.4 +
[lug_boot=big] * -12.66 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-37.2 +
[buying=vhigh] * -21.16 +
[buying=high] * -19.65 +
[buying=med] * 0.71 +
[buying=low] * 3.4 +
[maint=vhigh] * -23.6 +
[maint=high] * -18.14 +
[maint=med] * -6.13 +

[maint=low] * 23.82 +
[doors=2] * 20.61 +
[doors=3] * 0.62 +
[doors=4] * 0.68 +
[doors=5more] * 0.6 +
[persons=2] * -0.98 +
[persons=4] * 2.21 +
[lug_boot=small] * 19.15 +
[lug_boot=med] * 7.22 +
[lug_boot=big] * -22.2 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
2.67 +
[buying=vhigh] * -27.83 +
[buying=high] * -26.04 +
[buying=low] * 5.28 +
[maint=vhigh] * -45.54 +
[maint=high] * -42.52 +
[maint=med] * 7.04 +
[maint=low] * 1.76 +
[doors=2] * -28.57 +
[doors=3] * 1.29 +
[doors=4] * 1.36 +
[doors=5more] * 1.29 +
[persons=2] * -1.94 +
[persons=more] * 0.49 +
[lug_boot=small] * -40.74 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 20.66 +
[safety=high] * 13.39

LM_15:
Class unacc :
-81.79 +
[buying=vhigh] * 2.29 +
[buying=high] * 1.21 +
[buying=med] * -0.47 +
[buying=low] * -5.69 +
[maint=vhigh] * 5.82 +
[maint=high] * -13.14 +
[maint=med] * -0.68 +
[maint=low] * -0.79 +
[doors=2] * 76.94 +
[doors=3] * -0.01 +
[doors=5more] * -0.03 +
[persons=2] * 36.58 +
[persons=4] * -0.88 +
[lug_boot=small] * 69.14 +
[lug_boot=med] * 0.56 +
[lug_boot=big] * -0.41 +
[safety=low] * 12.31 +
[safety=high] * -2.11

Class acc :
-7.62 +
[buying=vhigh] * -1.37 +
[buying=high] * -0.12 +

[buying=med] * 1.63 +
[buying=low] * -3.65 +
[maint=vhigh] * 39.17 +
[maint=high] * 3.18 +
[maint=med] * -4.73 +
[maint=low] * -4.66 +
[doors=2] * 6.39 +
[doors=3] * -0.01 +
[doors=5more] * -0.02 +
[persons=2] * -3.71 +
[persons=4] * 0.32 +
[lug_boot=small] * 2.01 +
[lug_boot=med] * 5.69 +
[lug_boot=big] * -13.3 +
[safety=low] * -5.66 +
[safety=high] * 0.55

Class good :
-19.42 +
[buying=vhigh] * -21.16 +
[buying=high] * -19.65 +
[buying=med] * 0.71 +
[buying=low] * 3.4 +
[maint=vhigh] * -23.6 +
[maint=high] * -38.7 +
[maint=med] * 3.21 +
[maint=low] * 3.3 +
[doors=2] * 17.72 +
[doors=3] * 0.62 +
[doors=4] * 0.68 +
[doors=5more] * 0.6 +
[persons=2] * -0.98 +
[persons=4] * 2.21 +
[lug_boot=small] * 19.38 +
[lug_boot=med] * 11.05 +
[lug_boot=big] * -18.72 +
[safety=low] * -5.13 +
[safety=high] * 4.21

Class vgood :
6.18 +
[buying=vhigh] * -27.83 +
[buying=high] * -26.04 +
[buying=low] * 5.28 +
[maint=vhigh] * -59.34 +
[maint=high] * -6.53 +
[maint=med] * 0.49 +
[maint=low] * 0.83 +
[doors=2] * -35.79 +
[doors=3] * 1.29 +
[doors=4] * 1.36 +
[doors=5more] * 1.29 +
[persons=2] * -1.94 +
[persons=more] * 0.49 +
[lug_boot=small] * -49.4 +
[lug_boot=med] * 0.57 +
[lug_boot=big] * 24.13 +
[safety=high] * 13.39

Time taken to build model: 1.92 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	1707	98.7847 %
Incorrectly Classified Instances	21	1.2153 %
Kappa statistic	0.9734	
Mean absolute error	0.0065	
Root mean squared error	0.0758	
Relative absolute error	2.8462 %	
Root relative squared error	22.4183 %	
Total Number of Instances	1728	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,998	0,008	0,997	0,998	0,997	0,990	1,000	1,000	unacc
	0,966	0,006	0,979	0,966	0,972	0,965	0,995	0,981	acc
	0,942	0,004	0,903	0,942	0,922	0,919	0,991	0,848	good
	0,985	0,001	0,970	0,985	0,977	0,976	0,999	0,991	vgood
Weighted Avg.	0,988	0,007	0,988	0,988	0,988	0,981	0,998	0,989	

=== Confusion Matrix ===

a	b	c	d	<-- classified as
1207	3	0	0	a = unacc
4	371	7	2	b = acc
0	4	65	0	c = good
0	1	0	64	d = vgood