EBDS20102 - Supervised Machine Learning - *Assignment 2*

Predicting Supermarket Turnover using Demographic Factors

Mathijs de Jong 380891 Chao Liang 588140 Yuchou Peng 481487 Eva Mynott 495602

1 Introduction

Supermarket retailing has been witnessing dramatic changes in the last two decades due to the rapidly increasing market share of e-groceries. Customers are changing their shopping habits and are switching from the traditional offline shops to online stores to buy their daily supplies, particularly during the COVID-19 pandemic. To ensure the sustainability of the supermarket retailing business by optimally responding to changing consumer behaviours, it is inevitable for managers to recognize the demographic factors that historically affect yearly turnover. In this study, we analyze a cross-sectional data set on yearly turnover of groceries and demographic factors in Chicago areas. The latter consists of age group, ethnicity, family structure, income, living conditions, and shopper type. The aim of this research is to examine the demographic variables that assist in predicting yearly turnover of groceries. Our research question is as follows,

Which demographic factors are instrumental in predicting yearly supermarket turnover?

In view of the large number of potential relevant factors, we employ a linear regression model with an elastic net penalty on the parameters. Grid search and K-fold cross-validation are used to determine the optimal hyperparameters for this model. This paper is outlined as follows. Section 2 describes the data set briefly. Section 3 presents the methodology to answer the research question. Section 4 reports and interprets the empirical results of this study. Finally, we conclude our work in Section 5.

2 Data

The data set in this study is obtained from the University of Chicago Booth Kilts Center[1]. It contains cross-sectional data from 1996 on 77 Chicago area supermarkets, and corresponding demographic statistics for these areas. The demographic statistics are originally U.S. government data from 1990 for the Chicago metropolitan area. The variable of interest in this research is GROCERY_SUM, which is the total yearly turnover of groceries in U.S. dollars. After omitting five variables (STORE, ZIP, CITY, GROCCOUP_SUM and SHPINDX), the data set contains 44 independent variables that are used to predict supermarket turnover. These predictors contain information about consumer characteristics, such as age group, ethnicity, family structure, income group, living conditions, and shopper type. An exhaustive list of the variables in the data set and corresponding descriptions can be found in Table 1 in the Appendix. Table 1 in the Appendix also presents descriptive statistics. The response variable yearly supermarket turnover is 7.3 million U.S. dollar averaged over the Chicago areas in 1996, where the yearly turnover varies from 1.4 to 13.2 million U.S. dollar between areas. Moreover, we observe that on average Chicago households have a size of 2.7, and 20.4% of its citizens are non-white.

In addition, we analyse the correlations between the variables in the data set. In view of space limitations, this table is not disclosed but several insights are worth discussing. First, the response variable yearly supermarket turnover has a modest relationship with most predictor variables. It moderately comoves with income in an area and household value, but shows almost no relation with average household size or the fraction of people without a car. Moreover, we observe that a significant number of predictor variables show large comovement with at least one of the other predictor variables. For example, the fraction of children below the age of nine in an area is highly correlated with all household size statistics. Likewise, the log of income in an area is highly correlated with the fraction of college graduates, and strongly negatively correlated with poverty. This raises multicollinearity issues, and suggests that predictions based on straightforward multiple regression may lead to unstable prediction estimates. In other words, the correlation matrix vindicates the use of techniques such as regularized regression, which incorporates a penalizing function.

[1]Dominick's Dataset (2014). James M. Kilts Center, University of Chicago Booth School of Business. URL: `https://www.chicagobooth.edu/research/kilts/datasets/dominicks`

3 Methodology

In this section the methodology of this study is discussed. We consider a linear regression model with an elastic net penalty on the parameters to investigate the relation between the predictive variable and the collection of regressors. The relation can be modelled using a linear regression model which has the desirable property of returning parameter estimates of the individual effects of the regressors on the predictive variable. However, this approach is not suitable when one has a large set of regressors and a relatively low number of observations. In this case there is a high chance of encountering overfitting and multicollinearity. Shrinkage methods, such as the elastic net approach, can accommodate these problems by restricting the set of parameters. This allows one to study only the effects of relevant regressors, selected through the shrinkage approach, on the dependent variable.

Let $\mathcal{I} = \{1, 2, \ldots, n\}$ be the index set of the n observations and let $\mathcal{J} = \{1, 2, \ldots k\}$ be the index set of the k candidate regressors. The dependent variable is denoted by y_i and we assume the dependent variables to be independent and not identically distributed over $i \in \mathcal{I}$. Moreover, let $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,k})'$ denote the regressors (excluding the intercept), assumed to be non-stochastic for all $i \in \mathcal{I}$. The loss function of a linear regression model with shrinkage on the parameters is given by

$$\mathcal{L}(\boldsymbol{\beta}) = \frac{1}{n} \sum_{i \in \mathcal{I}} (y_i - \gamma - \mathbf{x}_i \boldsymbol{\beta})^2 + \lambda P(\boldsymbol{\beta}; \boldsymbol{\alpha}), \tag{1}$$

where the intercept γ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)'$ are the parameters to be estimated. Moreover, λ is the shrinkage scaling hyperparameter, $P : \mathbb{R}^k \to \mathbb{R}$ is some penalty function with hyperparameters $\boldsymbol{\alpha}$, all of which have to be selected by the researcher. Note that loss function (1) consists of the OLS loss function and a penalty term on the regression parameters. The elastic net penalty function, introduced by Zou and Hastie (2005), is given by $P(\boldsymbol{\beta}; \alpha) = \alpha \|\boldsymbol{\beta}\|_1 + (1 - \alpha)/2 \|\boldsymbol{\beta}\|_2^2$, where α is a scalar between zero and one. This penalty term is a weighted combination of the Ridge penalty term (Horel 1962) and the least absolute shrinkage and selection operator (LASSO) penalty term (Tibshirani 1996). The L_1-norm improves the prediction performance through variance reduction if the sample size is small relative to the number of explanatory variables and ensures higher interpretability due to simultaneous model selection. The L_2-norm shrinks large regression coefficients in order to reduce overfitting. Both terms help avoid computational problems of traditional regression methods in the case the number of explanatory variables is large relative to the number of observations. We minimize the loss function (1) to obtain estimates of $\hat{\boldsymbol{\beta}}$ using the analytical solution when it exists, that is, when α is zero. Otherwise, the Minimization by Majorization (MM) algorithm of Hunter and Lange (2004) is used for estimation.

The hyperparameters have to be chosen before the final analysis. We perform grid search and K-fold cross validation as described in Friedman, Hastie, and Tibshirani (2001). Grid search is an exhaustive search through a manually specified subset of the hyperparameter values. For each of the possible combinations of hyperparameters, the model is estimated and the combination yielding the best performance is selected. We consider 100 equally divided values in the interval $[0, 1]$ for alpha. For λ, we consider 0 and 99 values between 10^{-5} and 10^5.

The performance of a specific combination of hyperparameter values is approximated using K-fold cross validation. In this approach, the data is equally distributed over K folds. In each of K iterations, one of the folds serves as test set, whereas train coefficients are estimated on the data in the remaining $(K - 1)$ folds. The train coefficients are used to compute an out-of-sample metric on the test fold. We use the Root Mean Squared Error, that is,

$$RMSE_k = \sqrt{\sum_{i \in \mathcal{I}_k} \left(y_i - \mathbf{x}_i \hat{\boldsymbol{\beta}}\right)^2}, \tag{2}$$

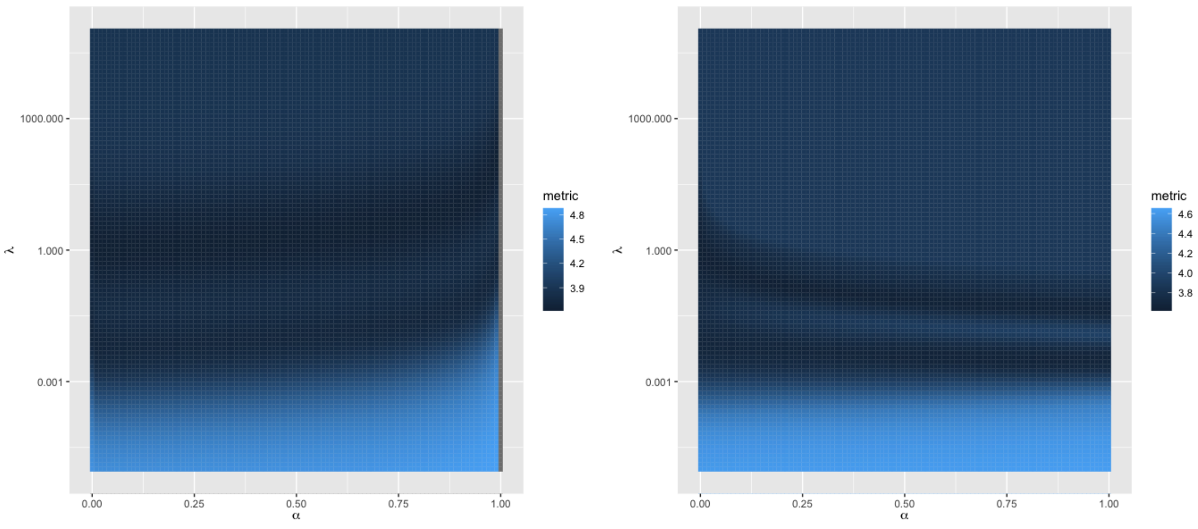
where $\mathcal{I}_k$ denotes the set of observations in the test set. After K iterations, the individual fold performances are combined using a summarizing metric, in our case we simply use the mean obtaining the mean RMSE of the K folds. We use 5 folds for the cross-validation step and divide the data into folds once at before applying grid search.

4 Results

This section presents the results of this study. In order to conclude which demographic factors shape predictions of the yearly supermarket turnover, we describe the results of a linear regression model with an elastic net penalty on the parameters. First the resulting optimal hyperparameters using 5-fold cross-validation are discussed. Second, the estimated coefficients from the final regression are examined.

Figure 1 presents two contour plots of the grid of hyperparameter combinations (λ, α). The hyperparameter λ is a positive penalty term, and α is a mixing parameter between ridge and LASSO (Groenen 2020). The parameter λ is varied from 0 and 1×10^{-5} to 1×10^{5}, whereas α ranges from 0 to 1. The metric used to obtain the optimal parameters is the root mean-squared error (RMSE). For each combination of parameter values, 5-fold cross-validation is performed to obtain an average RMSE over the folds. A dark blue shade represents a relatively small RMSE, whereas a light blue shade indicates a relatively large RMSE. Figure 1a shows the results for the implementation using the MM-algorithm, whereas Figure 1b displays the results obtained from the `glmnet` R package. Due to rounding errors in R, the MM-algorithm could not be used to compute values when α is equal to one (as it rounds it upwards resulting in a singular matrix that cannot be converted) and hence these are only shown using the `glmnet` package. A full overview of the optimal parameter estimates using the different approaches is given in Table 2. Although the plots are not completely similar, the direction of improvement is equivalent. A smaller value of the hyperparameter λ, moving towards one, decreases the root mean-squared error. In contrast, the α has a less pronounced effect on the average RMSE compared to λ.

Figure 1: Contour plot of the RMSE for grid of hyperparameters
This figure shows the root mean-squared error (RMSE) metric for a grid of penalty parameter λ values and mixing parameter α values. 5-fold cross-validation is used to obtain an average RMSE over the folds for all parameter values. Darker blue values indicate a lower, hence better, RMSE. Figure (a) presents the outcomes for the custom MM implication, whereas Figure (b) shows the results of the `glmnet` R package.



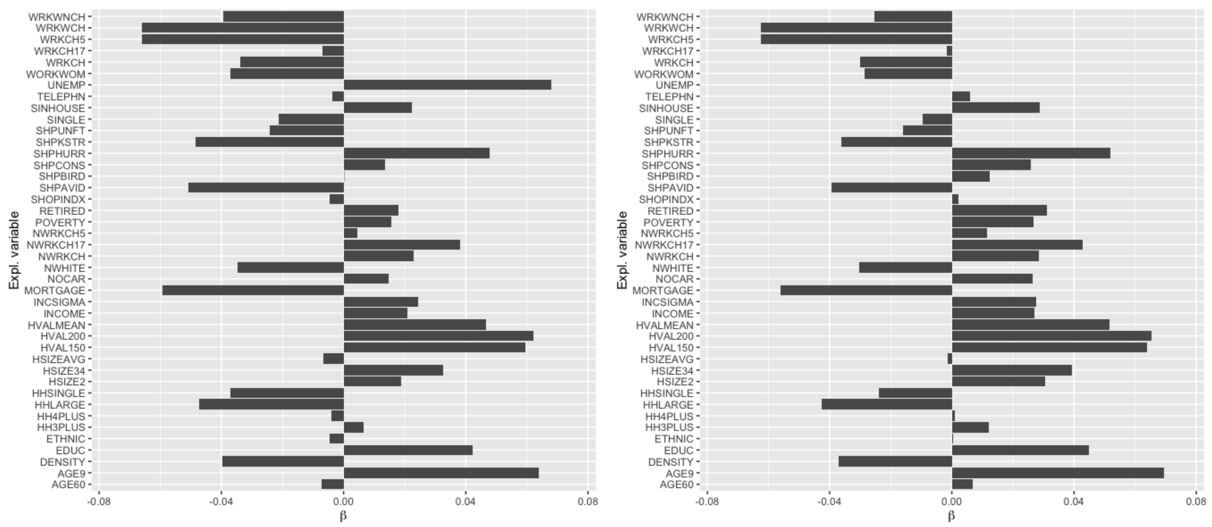
(a) MM-algorithm (b) `glmnet`

The optimal hyperparameters according to the implementation using the MM-algorithm are $\lambda = 1.264855$ and $\alpha = 0.171717$, whereas the optimal hyperparameters according to the `glmnet` R package are $\lambda = 1$ and $\alpha = 0$. In practical terms, the hyperparameter values of the MM implementation indicate that a combination of a LASSO and a ridge penalty is optimal, whereas the hyperparameter values of the `glmnet` R package indicate that a ridge penalty is optimal. The difference in hyperparameter estimates of the two methods are likely the results of either the `glmnet` implementation having a procedure that forces values to zero or it could be the result of using a significantly lower convergence tolerance when estimating beta, resulting in a large number of coefficients being equal to zero as shown in Table 5. Other factors that may contribute are rounding errors and different initial beta values.

Using the optimal cross-validated hyperparameters, we estimate the final regression, which either is an elastic net regression or a ridge regression. The regression weights are shown in Figure 2. For a non-zero penalty the coefficients become difficult to interpret, which is a cost of K-fold cross-validation. Due to the penalty term in the regression some coefficients are shrunk to zero (Groenen 2020). However, variables in our data set are often highly interdependent, so if one of the variables with a coefficient shrunk to zero changes, it may still have a large effect through the comovement with other variables that have large, non-zero coefficients.

Figure 2: Estimated Regression Coefficients

This figure shows the estimated regression coefficients for the optimal values of hyperparameters λ and α. The latter were determined using 5-fold cross-validation to obtain an average RMSE over the folds. The predictor variables are shown on the y-axis, whereas the estimated β's are displayed on the x-axis. Figure (a) shows the coefficients for the MM implementation, whereas Figure (b) shows the results using the `glmnet` R package.



(a) MM-algorithm

(b) `glmnet`

Despite the interpretation difficulty, we can still compare the coefficients in magnitude where a larger number indicates higher importance. The conjunction of the results from both methods indicates that household value, the fraction of people below age 9, and the fraction of hurried shoppers have a large positive impact on predicted supermarket turnover. For example, if the fraction of people with a household value of over 150K changes by one standard deviation, the predicted supermarket turnover increases by 0.06 standard deviations. In contrast, the fraction of working women with children below age 5, the fraction of people having a mortgage, and the fraction of avid shoppers have a significant negative effect on predicted supermarket turnover. Overall, average household size and ethnicity show almost no predictive power.

5 Conclusion

In this paper, we have considered an elastic net to predict the turnover of supermarkets based on a wide range of demographic variables. The cross-sectional data set used in this study contains data on 77 Chicago area supermarkets in 1996, and corresponding demographic statistics. All in all, we conclude that having a mortgage, working women with young children, and household value, are among the variables with the highest impact on predicted supermarket turnover. Ethnicity and average household size however, have almost no predictive power. A limitation of the current study is the data set that contains a small set of observations compared to a large number of predictors, which may lead to the problem of overfitting. This calls for a more recent, enriched data set on supermarket data and demographic statistics. Further research could then take into account the demographic changes over the past twenty-four years, by replicating our methodology on the more recent data.

References

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2001). *The elements of statistical learning.* Vol. 1. 10. Springer series in statistics New York.

Groenen, P.J.F. (2020). "Supervised Machine Learning Week 2". Unpublished University Lecture at Business Data Science/Erasmus University Rotterdam.

Horel, AE (1962). "Applications of Ridge Analysis Toregression Problems". In: *Chem. Eng. Progress.* 58, pp. 54–59.

Hunter, David R and Kenneth Lange (2004). "A tutorial on MM algorithms". In: *The American Statistician* 58.1, pp. 30–37.

Tibshirani, Robert (1996). "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1, pp. 267–288.

Zou, Hui and Trevor Hastie (2005). "Regularization and variable selection via the elastic net". In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2, pp. 301–320.

Appendix

Table 1: Supermarket Data Summary

This table reports the mean, standard deviation, first and third quantile, minimum and maximum, and unit of measurement for the cross-sectional data set from 1996 on Chicago area supermarket data and demographics. Note that the variables measured in fractions are percentages divided by 100, such that 0.025 in the table represents 2.5%.

		Descriptive statistics						
variable	n	mean	std. dev.	min	q_{25}	q_{75}	max	unit
grocery_sum	77	7,341,015	2,341,073	1,423,582	5,778,987	9,022,599	13,165,586	turnover $/year
age9	77	0.138	0.025	0.046	0.121	0.151	0.193	frac. age < 9
age60	77	0.173	0.063	0.058	0.122	0.214	0.307	frac. age > 60
ethnic	77	0.160	0.193	0.024	0.044	0.188	0.996	frac non-white
educ	77	0.227	0.114	0.050	0.146	0.284	0.528	frac. college grads
nocar	77	0.113	0.132	0.012	0.025	0.144	0.551	frac. without car
income	77	10.616	0.293	9.867	10.414	10.797	11.236	log of income
incsigma	77	24,841	2,295	20,360	23,488	26,458	30,277	std. of income
hsizeavg	77	2.665	0.263	1.554	2.543	2.790	3.309	avg. household size
hsize1	77	0.245	0.083	0.122	0.200	0.269	0.614	frac. household of 1
hsize2	77	0.309	0.031	0.219	0.290	0.333	0.369	frac. household of 2
hsize34	77	0.330	0.060	0.092	0.306	0.367	0.446	frac. household of 3/4
hsize567	77	0.116	0.031	0.014	0.098	0.132	0.216	frac. household of > 5
hh3plus	77	0.446	0.083	0.106	0.405	0.490	0.650	frac. household of > 3
hh4plus	77	0.274	0.063	0.041	0.241	0.305	0.443	frac. household of > 4
hhsingle	77	0.245	0.083	0.122	0.200	0.269	0.614	frac. detached house
hhlarge	77	0.116	0.031	0.014	0.098	0.132	0.216	frac. household of > 5
workwom	77	0.358	0.053	0.244	0.312	0.402	0.472	frac. working women
sinhouse	77	0.548	0.216	0.017	0.517	0.706	0.822	frac. single household
density	77	0.001	0.001	0.000	0.000	0.001	0.005	squared miles/capita
hval150	77	0.349	0.246	0.003	0.123	0.534	0.917	frac. value > 150K
hval200	77	0.186	0.186	0.001	0.043	0.268	0.781	frac. value > 200K
hvalmean	77	147.907	47.534	64.348	108.924	179.072	267.390	avg. household value $
single	77	0.280	0.068	0.203	0.242	0.286	0.593	frac. singles
retired	77	0.150	0.051	0.056	0.109	0.188	0.236	frac. retired
unemp	77	0.182	0.023	0.142	0.166	0.195	0.245	frac. unemployed
wrkch5	77	0.056	0.020	0.024	0.041	0.070	0.118	frac. emp. women with children age < 5
wrkch17	77	0.124	0.029	0.041	0.103	0.144	0.198	frac. emp. women with children age $6 - 17$
nwrkch5	77	0.084	0.028	0.030	0.064	0.101	0.169	frac. unemp. women with children age < 5
nwrkch17	77	0.070	0.021	0.018	0.059	0.082	0.122	frac. unemp. women with children age $6 - 17$
wrkch	77	0.180	0.044	0.071	0.149	0.214	0.293	frac. emp. women with children
nwrkch	77	0.154	0.043	0.048	0.123	0.183	0.250	frac. unemp. women with children
wrkwch	77	0.055	0.020	0.024	0.041	0.069	0.115	frac. emp. women with children age < 5
wrkwnch	77	0.258	0.044	0.157	0.227	0.282	0.460	frac. emp. women with no children
telephn	77	0.977	0.029	0.839	0.976	0.993	0.998	frac. households with telephones
mortgage	77	0.710	0.147	0.443	0.617	0.826	0.960	frac. households with mortgages
nwhite	77	0.204	0.194	0.035	0.091	0.205	0.995	frac. non-white
poverty	77	0.058	0.045	0.014	0.027	0.076	0.213	frac. income $<$ \$15K
shopcons	77	0.082	0.062	0.019	0.037	0.115	0.279	frac. constrained shoppers
shophurr	77	0.153	0.059	0.026	0.110	0.191	0.286	frac. hurried shoppers
shopavid	77	0.189	0.043	0.061	0.161	0.220	0.310	frac. avid shoppers
shopstr	77	0.284	0.066	0.184	0.232	0.330	0.558	frac. shopping stranges
shopunft	77	0.246	0.055	0.145	0.197	0.291	0.391	frac. unfettered shoppers
shopbird	77	0.046	0.025	0.004	0.025	0.064	0.105	frac. shopper birds
shopindx	77	0.736	0.246	0.000	0.730	0.890	0.986	frac. ability to shop

Parameter estimates comparison

Table 2: Parameter Estimates Comparison
This table reports parameter estimates using the MM-algorithm and the `glmnet` package for the optimal values of α and λ according to the two methods.

	MM-alg.	MM-alg.	`glmnet`	`glmnet`
α	0.171717	0	0.171717	0
λ	1.264855	1	1.264855	1
Parameter estimates				
`Intercept`	0.000000	0.000000	0.000000	0.000000
`AGE60`	-0.007137	0.010511	0.000000	0.006820
`AGE9`	0.064013	0.118967	0.000000	0.069308
`DENSITY`	-0.039711	-0.065376	0.000000	-0.037167
`EDUC`	0.042191	0.047525	0.000000	0.044717
`ETHNIC`	-0.004628	0.006676	0.000000	0.000390
`HH3PLUS`	0.006596	0.018533	0.000000	0.012033
`HH4PLUS`	-0.003931	-0.002468	0.000000	0.001112
`HHLARGE`	-0.047339	-0.074516	0.000000	-0.042563
`HHSINGLE`	-0.037085	-0.036778	0.000000	-0.023891
`HSIZE2`	0.018814	0.048816	0.000000	0.030667
`HSIZE34`	0.032456	0.064396	0.000000	0.039321
`HSIZEAVG`	-0.006563	-0.005564	0.000000	-0.001484
`HVAL150`	0.059370	0.082381	0.015432	0.063915
`HVAL200`	0.062249	0.072052	0.037757	0.065226
`HVALMEAN`	0.046656	0.057843	0.003803	0.051449
`INCOME`	0.020887	0.027713	0.000000	0.027108
`INCSIGMA`	0.024486	0.012163	0.031715	0.027613
`MORTGAGE`	-0.059370	-0.093061	0.000000	-0.056114
`NOCAR`	0.014817	0.046031	0.000000	0.026412
`NWHITE`	-0.034745	-0.051542	0.000000	-0.030344
`NWRKCH`	0.022803	0.033222	0.000000	0.028614
`NWRKCH17`	0.038045	0.039106	0.000000	0.042732
`NWRKCH5`	0.004547	0.021872	0.000000	0.011640
`POVERTY`	0.015578	0.044645	0.000000	0.026861
`RETIRED`	0.017876	0.038782	0.000000	0.030967
`SHOPINDX`	-0.004695	0.002341	0.000000	0.002142
`SHPAVID`	-0.050809	-0.054833	0.000000	-0.039520
`SHPBIRD`	0.000132	-0.000893	0.000000	0.012459
`SHPCONS`	0.013550	0.043352	0.000000	0.025878
`SHPHURR`	0.047924	0.076358	0.000000	0.051857
`SHPKSTR`	-0.048540	-0.038299	0.000000	-0.036181
`SHPUNFT`	-0.024201	-0.041141	0.000000	-0.015881
`SINGLE`	-0.021164	-0.006930	0.000000	-0.009531
`SINHOUSE`	0.022230	0.046428	0.000000	0.028764
`TELEPHN`	-0.003699	0.021507	0.000000	0.005909
`UNEMP`	0.068032	0.116889	0.000000	0.076901
`WORKWOM`	-0.036963	-0.021917	0.000000	-0.028558
`WRKCH`	-0.033883	-0.034166	0.000000	-0.029887
`WRKCH17`	-0.006995	0.010315	0.000000	-0.001583
`WRKCH5`	-0.066064	-0.088921	-0.003938	-0.062385
`WRKWCH`	-0.066137	-0.088845	-0.004761	-0.062462
`WRKWNCH`	-0.039261	-0.031063	0.000000	-0.025193

R Code

In this section, we show the code used to generate the results in this study. The code extends the previous implementations in the course and some of the methods referred in the functions specifically implemented for this study rely on the base code shown in the next code snippet.

```r
################################################################################
# Helper functions for computing summary statistics for linear regression
# models.
#
# Inputs:
#   beta:          Vector of parameter estimates.
#   x:             Table containing numerical explanatory variables.
#   y:             Column containing a numerical dependent variable.
#
# Output:
#   Summary statistic for given linear regression model.
rss = function(beta, x, y) sum((y - x %*% beta) ^ 2)
r2 = function(beta, x, y) 1 - rss(beta, x, y) / sum((y - mean(y)) ^ 2)
adj.r2 = function(beta, x, y) {
  N = nrow(x); return(1 - (1 - r2(beta, x, y)) * (N - 1) / (N - sum(beta != 0)))
}

################################################################################
# Helper functions for initializing machine learning models.
#
# Inputs:
#   beta.init:     Initial beta parameter.
#   x:             Table containing numerical explanatory variables.
#   y:             Column containing a numerical dependent variable.
#   intercept:     Indicator for whether to include an intercept or not.
#   standardize:   Indicator for whether to standardize the input. Ignored when
#                  intercept is TRUE.
#
# Output:
#   Dependent or explanatory variables formatted to be used in a regression
#   model.
create.x = function(x, intercept, standardize) {
  x = data.matrix(x)
  if (intercept) return(cbind(1, x)) else if (standardize) x = scale(x)
  return(x)
}
create.y = function(y, intercept, standardize) {
  y = data.matrix(y)
  if (!intercept & standardize) y = scale(y)
  return(y)
}
initialize.beta = function(beta.init, x) {
  if(is.null(beta.init)) return(2 * runif(ncol(x)) - 1)
  return(beta.init)
}

################################################################################
# Helper functions for computing the loss of a linear regression model with an
# elastic net penalty on the parameter estimates.
```

```r
#
# Inputs:
#   x:             Table containing numerical explanatory variables.
#   y:             Column containing a numerical dependent variable.
#   intercept:     Indicator for whether to include an intercept or not.
#   lambda.l1:     Penalty term for the L1-norm term.
#   lambda.l2:     Penalty term for the L2-norm term.
#
# Output:
#   Loss of the given linear model with an elastic net penalty.
elastic.net.loss = function(beta, x, y, intercept, lambda.l1, lambda.l2)
  rss(beta, x, y) / (2 * nrow(x)) +
  lambda.l1 * sum(abs(beta[(1 + intercept):ncol(x)])) +
  lambda.l2 * sum(beta[(1 + intercept):ncol(x)] ^ 2) / 2

################################################################################
# Helper function for descaling a linear regression estimator estimated on
# scaled data.
#
# Inputs:
#   beta:          Vector of parameter estimates.
#   x:             Table containing unscaled numerical explanatory variables.
#   y:             Column containing an unscaled numerical dependent variable.
#
# Output:
#   Descaled linear regression estimator.
descale.beta = function(beta, x, y) {
  beta = sd(y) * beta / apply(x, 2, sd)
  beta = c(mean(y) - sum(colMeans(x) * beta), beta)
  names(beta) = c('(Intercept)', colnames(x))
  return(beta)
}

################################################################################
# Helper functions for displaying progress in custom implementation of machine
# learning models.
progress.str = function(line.list) {
  for (l in line.list) cat(progress.line(l)); cat('\n\n')
}

progress.line = function(line) paste0(
  format(paste0(line[1], ':'), width=25), format(line[2], width=25,
    justify='right', nsmall=ifelse(is.integer(line[2]), 0, 10)),'\n')
```

For grid search K-fold cross-validation, we implemented a separate function. Note that this function is also used to construct the graphs in this report.

```r
grid.search.cross.validation = function(x, y, estimator, params.list,
  n.folds=10, ind.metric, comb.metric, fold.id=NULL, force=T, verbose=F,
  heatmap=F, heat.scale=NULL, plot.coef=F, ...) {
  # Implementation of hyperparameter tuning using grid search using K-fold
  # cross-validation for given data, a given estimator, and given metrics.
  #
  # Inputs:
```

```
#   x:           Table containing numerical explanatory variables.
#   y:           Column containing a numerical dependent variable.
#   estimator:   Estimator to use in hyperparameter tuning.
#   n.folds:     Number of folds - default is 10. Although n.folds can be as
#                large as the sample size (leave-one-out CV), it is not
#                recommended for large datasets.
#   fold.id:     An optional vector of values between 1 and n.fold
#                identifying what fold each observation is in. If supplied,
#                n.fold can be missing.
#   ind.metric:  Metric for evaluating performance on fold.
#   comb.metric: Combination function for individual metrics.
#   force:       Indicator whether not to terminate when encountering errors.
#                Default is T, that is, errors are skipped with warning.
#   verbose:     Indicator for displaying progress bar. Default is FALSE.
#   heatmap:     Indicator for whether or not to display a heatmap of the
#                gridsearch outcomes.
#   heat.scale:  Optional argument to specify the scale used in the heatmap.
#                Default is NULL.
#   plot.coef:   Indicator for whether or not to display optimal coefficients.
#   ...:         Additional arguments that can be passed to the estimator.
#
# Output:
#   Dataframe containing the results of the linear regression model with
#   elastic net penalty term solved using the MM algorithm.

# Install and load dependencies
while (!require('dplyr')) install.packages('dplyr', quiet=T)
while (!require('ggplot2')) install.packages('ggplot2', quiet=T)
while (!require('latex2exp')) install.packages('latex2exp')

# Define constants
y = data.matrix(y); x = data.matrix(x); N = nrow(x); b.metric = Inf

# Specify fold ids if not given and initialize metrics and ids vector
if(is.null(fold.id)) fold.id = ((1:N) %% n.folds + 1)[sample(N, N)]
else n.folds =length(unique(fold.id))
metrics = rep(NULL, n.folds); test.ids = matrix(nrow=n.folds, ncol=N)
for (fold in 1:n.folds) test.ids[fold, ] = (fold.id == fold)

# Create grid for cross validation search
grid = expand.grid(params.list)
n.combs = nrow(grid); metric = rep(NULL, n.combs)

# If verbose, initialize progress bar
if (verbose) pb = dplyr::progress_estimated(n.combs * n.folds)

# Apply grid search
for (i in 1:n.combs) {

  # Apply cross validation and if verbose, update progress bar
  for (fold in 1:n.folds) { if (verbose) pb$tick()$print()

    # Compute individual metric on fold
```

```r
      metrics[fold] = tryCatch(
        ind.metric(do.call(estimator, c(list(x=x[!test.ids[fold, ], ],
          y=y[!test.ids[fold, ]]), as.list(grid[i, ]), list(...)))$beta,
          x[test.ids[fold, ], ], y[test.ids[fold, ]]),
        error = function(e) {
          warning(paste('Failed for', paste(names(params.list), '=', grid[i, ],
            collapse=', ')))
          if (force & grepl('.*singular.*', e$message)) return(Inf)
          stop(e)
        }
      )
    }

    # Combine performances on folds to overall performance
    metric[i] = comb.metric(metrics)
  }

  # Extract optimal hyperparameters
  b.id = which.min(metric); b.metric = metric[b.id]; b.params = grid[b.id, ]

  # Plot heatmaps if required
  combs = combn(names(params.list), 2); grid$metric = metric
  if (heatmap) for (i in 1:ncol(combs)) {
    col.x = combs[1, i]; col.y = combs[2, i]
    p = ggplot(data = grid, aes_string(x=col.x, y=col.y)) + geom_tile(aes(
      color=metric, fill=metric)) + ylab(TeX(paste0('$\\', col.y, '$'))) +
      xlab(TeX(paste0('$\\', col.x, '$')))
    if (!is.null(heat.scale))
      p = p + scale_y_continuous(trans=heat.scale[col.y]) +
      scale_x_continuous(trans=heat.scale[col.x])
    print(p)
  }

  # Estimate best beta and if required, plot outcomes
  best.b = do.call(estimator, c(list(x=x, y=y), as.list(b.params),
    list(...)))$beta
  if (plot.coef) print(ggplot(data.frame(y=colnames(x), b=as.vector(best.b)),
    aes(b, y)) + geom_col() + ylab('Expl. variable') + xlab(TeX('$\\beta$')) +
    xlim(-0.075, 0.075))

  # Reformat optimal hyperparameters
  if (length(params.list) > 1) b.params = c(b.params)

  return(list(
    'beta' = best.b,
    'params' = b.params,
    'metric' = b.metric
  ))
}
```

The following function is the linear regression model with an elastic net penalty term on the parameters that is estimated using the MM-algorithm. Note that this function calls the linear model with a Ridge penalty term when α is equal to zero.

```r
elastic.net.lm = function(x, y, lambda, alpha, intercept=F, standardize=T,
  beta.init=NULL, beta.tol=0, loss.tol=1e-6, eps=1e-6, seed=NULL, verbose=0) {
  # Implementation of the MM algorithm solver for a linear regression model
  # an elastic net penalty term.
  #
  # Inputs:
  #   x:            Table containing numerical explanatory variables.
  #   y:            Column containing a numerical dependent variable.
  #   lambda:       Penalty scaling constant.
  #   alpha:        Scalar of penalty for L1-norm of beta. Note that the scalar
  #                 assigned to the L2-norm is equal to (1 - alpha) / 2.
  #   intercept:    Indicator for whether or not to add an intercept. Default
  #                 is FALSE. If TRUE, standardize is ignored.
  #   standardize:  Indicator for whether or not to scale data. If intercept is
  #                 TRUE, this argument is ignored. Default is TRUE.
  #   beta.init:    Optional initial value of betas. If NULL, a random beta is
  #                 sampled from the continuous uniform distribution on the
  #                 interval [0, 1]. Default is NULL.
  #   beta.tol:     Rounding tolerance for beta. Default is 0.
  #   loss.tol:     Tolerated loss, default is 1e-6.
  #   eps:          Correcting value in computation of D matrix, default is
  #                 1e-6.
  #   seed:         Optional seed, used for generating the random initial beta.
  #                 Ignored when an initial beta is provided. Default is NULL.
  #   verbose:      Integer indicating the step-size of printing progress
  #                 updates, default is 0, that is, no progress updates.
  #
  # Output:
  #   Dataframe containing the results of the linear regression model with
  #   elastic net penalty term solved using the MM algorithm.

  # Estimate model with Ridge regression if possible
  if (alpha == 0) return(ridge.lm(x, y, lambda, intercept, standardize,
    beta.tol, verbose))

  # Import our own shared own functions
  source('../../base.R'); descale = function(beta) descale.beta(beta, x, y)

  # Add intercept or standarize data if necessary
  x = create.x(x, intercept, standardize)
  y = create.y(y, intercept, standardize)

  # Define constants
  N = nrow(x); P = ncol(x); if (!is.null(seed)) set.seed(seed)
  lambda.l1 = lambda * alpha; lambda.l2 = lambda * (1 - alpha)
  lambda.l2.I = diag(rep(N * lambda.l2, P)); if (intercept) lambda.l2.I[1,1] = 0
  Xt.X = crossprod(x); Xt.y = crossprod(x, y)

  # Define helper functions for computing specific expressions and loss
  lambda.l1.D = function(beta) {
    beta[abs(beta) < eps] = eps
    lambda.l1.D = diag(N * lambda.l1 / abs(beta))
    if (intercept) lambda.l1.D[1,1] = 0
    return(lambda.l1.D)
  }
  loss = function(beta)
    elastic.net.loss(beta, x, y, intercept, lambda.l1, lambda.l2)
  pline = function(i, o, n, d)
    list(c('Iteration', i), c('Loss.old', o), c('Loss.new', n), c('Delta', d))

  # Choose some inital beta_0 and compute initial loss
  b.new = initialize.beta(beta.init, x); l.new = loss(b.new)

  # Update iteration and replace old parameters by previous until convergence
  i = 0L; while (TRUE) { i = i + 1L; l.old = l.new; b.old = b.new

    # Update parameters, loss and delta
    b.new = solve(Xt.X + lambda.l1.D(b.old) + lambda.l2.I, Xt.y)
    l.new = loss(b.new); diff = l.old - l.new

    # Display progress if verbose
    if (verbose & (i %% verbose == 0)) progress(pline(i, l.new, l.old, diff))

    # Break if improvement smaller than tol, that is, sufficient convergence
    if (diff / l.old < loss.tol) break
  }

  # Ensure information of last iteration is displayed
  if (verbose & (i %% verbose)) progress(pline(i, l.new, l.old, diff))

  # Force elements smaller than beta.tol to zero
  b.new[abs(b.new) < beta.tol] = 0

  # Descale beta if necessary
  if (intercept | !standardize) beta = c(0, b.new) else beta = descale(b.new)

  return(list('a0'=beta[1], 'beta'=beta[-1], 'alpha'=alpha, 'lambda'=lambda,
    'loss'=loss(b.new), 'R^2'=r2(b.new, x, y),
    'adjusted R^2'=adj.r2(b.new, x, y)))
}
```

The following function is used for estimating the linear Ridge regression model. Note that it uses the analytical solution to find the parameter estimates.

```
ridge.lm = function(x, y, lambda, intercept=F, standardize=T, descale=T,
  beta.tol=0, verbose=0) {
  # Implementation of the analytical solution for the linear regression model
  # with a Ridge penalty term.
  #
  # Inputs:
  #   x:            Table containing numerical explanatory variables.
  #   y:            Column containing a numerical dependent variable.
  #   lambda:       Penalty scaling constant.
  #   intercept:    Indicator for whether or not to add an intercept. Default
  #                 is FALSE. If TRUE, standardize is ignored.
  #   standardize:  Indicator for whether or not to scale data. Default is TRUE.
  #   beta.tol:     Rounding tolerance for beta, default is 0.
```

```r
  #   verbose:      Integer indicating the step-size of printing progress
  #                 updates, default is 0, that is, no progress updates.
  #
  # Output:
  #   Dataframe containing the results of the linear regression model with
  #   Ridge penalty term solved using the analytical solution

  # Import our own shared own functions
  source('../../base.R'); descale = function(beta) descale.beta(beta, x, y)

  # Add intercept or standarize data if necessary
  x = create.x(x, intercept, standardize)
  y = create.y(y, intercept, standardize)

  # Derive beta estimate
  b.new = solve(crossprod(x) + nrow(x) * lambda / 2 * diag(ncol(x)),
    crossprod(x, y))

  # Set elements smaller than beta.tol to zero
  b.new[abs(b.new) < beta.tol] = 0

  # Descale beta if necessary
  if (intercept | !standardize) beta = c(0, b.new) else beta = descale(b.new)

  return(list('a0'=beta[1], 'beta'=beta[-1], 'alpha'=0, 'lambda'=lambda,
    'loss'=elastic.net.loss(b.new, x, y, intercept, lambda.l1=0, lambda),
    'R^2'=r2(b.new, x, y), 'adjusted R^2'=adj.r2(b.new, x, y)))
}
```

Finally, the following code was used to generate the results, tables, and figures in this report.

```r
################################################################################
# Initialize local settings
################################################################################

# Specify options
options(scipen=999)
set.seed(42)

################################################################################
# Load dependencies
################################################################################

# Install and load packages
while (!require('glmnet')) install.packages('glmnet', quiet=T)
while (!require('SVMMaj')) install.packages('SVMMaj', quiet=T)
while (!require('xtable')) install.packages('xtable', quiet=T)

# Load dependencies
source('../../base.R')
source('elastic.net.lm.R')
source('grid.search.cross.validation.R')
source('ridge.lm.R')
```

```r
################################################################################
# Pre-process data
################################################################################

# Load data
df = SVMMaj::supermarket1996; df = df[sort(colnames(df))]
df = subset(df, select=-c(CITY, GROCCOUP_sum, SHPINDX, STORE, ZIP))

# Define dependent and independent variables
dep.var = 'GROCERY_sum'; y = df[dep.var]; x = df[colnames(df) != dep.var]

# OPTIONAL: Remove duplicate columns
x = x[, -which(duplicated(t(x)))]

# Specify hyperparameter values to consider
params.length = 100; params.list = list(
  'alpha' = seq(0, 1, length.out=params.length),
  'lambda' = c(0, 10 ^ seq(-5, 5, length.out=params.length - 1))
)
heat.scale = c('alpha'='identity', 'lambda'='log10')

# Specify fold ids
N = nrow(x); n.folds = 5; fold.id = ((1:N) %% n.folds + 1)[sample(N, N)]

################################################################################
# Hyperparameter tuning
################################################################################

# Define metric functions
lm.rmse = function(beta, x, y) sqrt(mean(sum((y - x %*% beta) ^ 2)))

# Hyperparameter tuning using estimator based on MM algorithm
gscv.own = grid.search.cross.validation(scale(x), scale(y), elastic.net.lm,
  params.list, ind.metric=lm.rmse, comb.metric=mean, fold.id=fold.id,
  verbose=T, force=T, heatmap=T, heat.scale=heat.scale, plot.coef=T,
  intercept=F, standardize=F)
progress.str(list(c('Alpha', gscv.own$params$alpha), c('Lambda',
  gscv.own$params$lambda)))

# Hyperparameter tuning using glmnet estimator
gscv.glm = grid.search.cross.validation(scale(x), scale(y), glmnet,
  params.list, ind.metric=lm.rmse, comb.metric=mean, fold.id=fold.id,
  verbose=T, force=T, heatmap=T, heat.scale=heat.scale, plot.coef=T,
  intercept=F, standardize=F)
progress.str(list(c('Alpha', gscv.glm$params$alpha), c('Lambda',
  gscv.glm$params$lambda)))


################################################################################
# Generate results
################################################################################

# Estimate on all data for optimal values of lambda and alpha
```

```r
res.own.own = elastic.net.lm(scale(x), scale(y), alpha=gscv.own$params$alpha,
  lambda=gscv.own$params$lambda, standardize=F)
res.own.glm = elastic.net.lm(scale(x), scale(y), alpha=gscv.glm$params$alpha,
  lambda=gscv.glm$params$lambda, standardize=F)
res.glm.own = glmnet(scale(x), scale(y), alpha=gscv.own$params$alpha,
  lambda=gscv.own$params$lambda, standardize=F)
res.glm.glm = glmnet(scale(x), scale(y), alpha=gscv.glm$params$alpha,
  lambda=gscv.glm$params$lambda, standardize=F)

# Compare implementation of MM algorithm to glmnet outcome
res.table = data.frame(
  'own.own' = c(res.own.own$a0, res.own.own$beta),
  'own.glm' = c(res.own.glm$a0, res.own.glm$beta),
  'glm.own' = c(res.glm.own$a0, as.vector(res.glm.own$beta)),
  'glm.glm' = c(res.glm.glm$a0, as.vector(res.glm.glm$beta))
)
rownames(res.table) = paste0('\\texttt{', c('Intercept', colnames(x)), '}')

# Export table to LaTeX
xtable(res.table, digits=6)
```