3 Problem 3

In this part we will explore a data set containing sales data of orange juice. The goal is to predict the Purchase variable using various machine learning classification methods. We have data on two types of orange juice, Citrus Hill (CH) and Minute Maid (MM) and will build a few models to predict which product a consumer might purchase.

3.1 Data Splitting

We begin by splitting our data into a reproducible training- and test set.

```r
set.seed(1998) # for reproducibility
# Randomly select 800 indices for training
train_indices <- sample(nrow(oj_data), 800)
train_data <- oj_data[train_indices, ]
test_data <- oj_data[-train_indices, ]
```

3.2 Model and Method Training with Evaluation

We will now explore and compare different suitable models that could adequately predict our dependent variable of interest based on a few independent variables e.g. customer brand loyalty or price charged for each product.

3.2.1 Feature selection for model comparison

Since we are going to compare many models that are all built upon different underlying assumptions we will need a consistent and fair approach to feature selection. Of course these models would perform better if we did feature selection in each model building phase but for simplicity and interpretability we will create these feature subsets now.

Let's go with a singular additive model approach, or in this case perhaps grouped additive approach would be a better name, and step wise add features to make four sets. It is also worth mentioning that we removed features that caused multicollinearity before the feature selection.

```r
price_features<-c("PriceCH","PriceMM")
discount_features<-c("DiscCH","DiscMM","PctDiscCH","PctDiscMM")
store_loyalty_features<-c("StoreID","STORE","LoyalCH")
promo_temporal_features<-c("SpecialCH","SpecialMM","WeekofPurchase")
```

We start with a basic price feature model then we add discount and sale price features. After that we incorporate location and loyalty factors and lastly add promotional and temporal aspects to the feature set.

3.2.2 Model building - Generalized setup

Now that we have created our feature subsets all of the subsequent models will be built on the same basis: initialize a list, loop through each feature set and fit the desired model. Here is the example for logistic regression.

```
models <- list()
for (model_name in names(feature_sets)) {
formula <- as.formula(
paste("Purchase ~", paste(feature_sets[[model_name]], collapse=" + ")))
models[[model_name]] <-
glm(formula, data=train_data, family=binomial()) }
```

We then need to use our models in order to predict purchases and compare the prediction accuracy of the models on the test set for out of sample accuracy testing (or training set for in-sample testing).

```
binary_predictions <- list()
confusion_matrices <- list()
for (model_name in names(models)) {
  predicted_probs <-
  predict(models[[model_name]],newdata=test_data,type="response")
  binary_predictions[[model_name]] <- ifelse(predicted_probs>0.5,1,0)
  confusion_matrices[[model_name]] <-
  table(Predicted = binary_predictions[[model_name]],
  Actual = test_data$Purchase)}
```

The confusion matrix is of course our most valuable tool for testing in classification. After creating the confusion matrices for comparison the next step is looking at our accuracy for each model. Which we can either calculate from our confusion matrix or comparing predicted values with actual values by mean. Other metrics like specificity or sensitivity i.a. could also be calculated if desired. The accuracy is found from the diagonal of the confusion matrix relative to the whole:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

3.2.3 Model and method comparison of accuracy

	LR'	LR*	LDA'	LDA*	QDA'	QDA*	NB*	KNN	KNN^{CV}
M1	65.00	54.07	65.00	54.07	65.50	55.93	55.93	61.11	59.63
M2	67.50	60.37	67.12	59.26	66.00	62.96	64.81	60.74	60.00
M3	82.50	86.30	82.25	85.93	81.00	83.33	84.07	82.59	80.37
M4	82.50	86.30	82.50	85.93	80.50	84.81	80.00	70.37	73.33

Table 1: * methods are out of sample testing (OOS) while ' are in-sample testing; KNN built with 10 fold Cross-Validation (CV) tuning k from 1-20 see Appendix 5.2

According to our accuracy findings in table 1 a logistic regression method performs the best on the selected sub features for OOS testing. We see higher accuracy over all OOS than in-sample which we find rather surprising. This is only the case for models 3 and 4 though that contain some of our most important features (e.g. PriceMM and LoyalCH) which shows that the first two models suffer somewhat from OVB. It's also interesting to see how adding promotional and temporal features has very little effect on our logistic regression- and discriminant analysis methods but a very high one on Naive Bayes and KNN. The effect of promotional features, which are binary, should not affect KNN a lot but adding the temporal effect (week of purchase) could skew our distance calculations since we did no scaling on that feature (or for any feature here for that matter). As for Naive Bayes it can be used for mixed features but it assumes independence between said features therefore adding more features will alter probability estimations. Linear methods can therefore often be more robust in this regard as we see in our results here.

3.3 SVM

We will now use Support Vector Machine (SVM) classification in a two feature space using the same sample split as before for consistency. First we'll need to select the two features. We could use the two most important features in the linear model but since we might get non-linear results for our kernel with CV it's perhaps better to use more complex feature selection methods. After trying recursive feature selection (RFE) and deeming it overly complex for this situation we opted to use the two most prominent features selected by a linear subset selection: Customer brand loyalty for CH (LoyalCH) and Discount offered for MM (DiscMM) found by forwards subset selection. This should ensure at least some geometric seperation to perform SVM classification.

We will look at SVM with both a radial- and polynomial kernel selected by CV to find: c, γ and the degrees of the polynomial. Then we test our model the same way as before. After testing we found the polynomial kernel to perform better as calculated by CV in appendix 5.3.

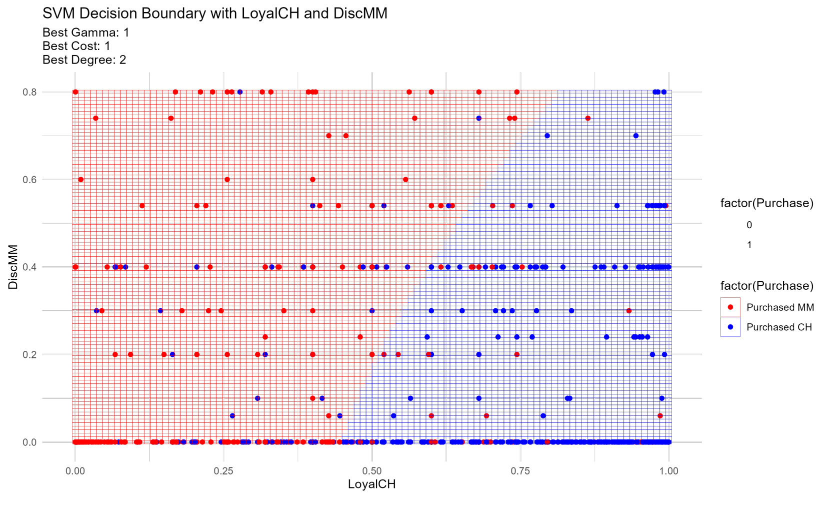

Figure 2: Polynomial kernel

Although a radial kernel and a polynomial one look similar there is a difference on the accuracy of the testing set as we go from about 81% to 84% accuracy on the testing set or from 49 wrong readings to 42. [1] A polynomial kernel gives us a more even distribution of false positives and false negatives (balance) while a radial one skews to more false positive results. The accuracy increase is not too surprising since we use fewer tuning parameters for our kernel estimation (324 vs. 2560 parameters).

3.4 Tree based methods

Now let's explore how multiple tree based methods handle our data set. We will look at: bagging, random forests and boosting models (using CV to determine the number of trees in boosting). Since we either average out predictions or build them sequentially traditional pruning is not necessary. We start by fitting the models and then predicting on the test set to calculate our accuracy- and error rates (reciprocal).

```
set.seed(1998)
tree_models <- list(
  bagging = randomForest(Purchase ~ ., data = train_data,
  mtry = ncol(train_data) / 3),
  random_forest = randomForest(Purchase ~ ., data = train_data),
  boosting = train(
    Purchase ~ ., data = train_data,
    method = "gbm",
    trControl = trainControl(method = "cv"),
    verbose = FALSE))

predictions <- lapply(tree_models,
                      function(model) predict(model, test_data))
```

	Bagging	Random Forest	Boosting
Accuracy rate (%)	83.33	83.70	84.44
Error rate (%)	16.67	16.30	15.56

Table 2: Accuracy rate of different tree based methods

As we can see in our Accuracy results of the tree-based methods they will perform about as well as Naive Bayes or Quadratic Discriminant analysis in Table 1.

We can now see that after exploring a few classification based methods that for this particular data set and sample split we can predict the purchase of orange juice between CH and MM with around 80% to 85% accuracy rate using various methods and models.

[1]See Appendix 5.4