MATH 3190 Homework 8

Focus: Notes 10 and Notes 11

Due April 25, 2024

Your homework should be completed in R Markdown or Quarto and Knitted to an html or pdf document. You will "turn in" this homework by uploading to your GitHub Math_3190_Assignment repository in the Homework directory.

Problem 1 - Time Series (34 points)

In the `itsmr` package is the dataset called `wine` that contains 142 accounts of Australian red wine sales per month from January 1980 to October 1991.

Part a (3 points)

Load in this data and convert it to a `tsibble`. You can do this by first making it a `ts` object by typing `ts(wine, deltat = 1/12, start = 1980)` and then piping that into the `as_tsibble()` function in the `tsibble` package.

Then use `ggplot()` to plot the time series. Show both the points and lines connecting them. On the x axis, put each year using the `scale_x_yearmonth()` function and give the plot a title.

Part b (1 point)

Remove the last 34 rows from the wine data. We will use them to compare how our forecasts look later.

Part c (3 points)

Use the `gg_season()` plot in the `feasts` library to plot the seasonality. Does it seem like there is trend and/or seasonality in this time series? If so, explain the patterns.

Part d (3 points)

Use the `adf.test()` function in the `tseries` library to perform a Dickey-Fuller test to determine if we have evidence this time series is stationary. Only the column with the sales should be entered. Since we have data by month, use the `k = 12` option. Comment on the results of the test.

Part e (4 points)

Difference the data once. Take $Y_t = Y_{t+1} - Y_t$ for $t = 1, \ldots, n-1$. In the notes, I defined the differences as $Y_t = Y_t - Y_{t+1}$, but it actually does not matter. **Plot** this differenced time series (you can just put the index on the x-axis) and use the `adf.test()` function again on the differenced data. Again, use `k = 12` option. **Comment** on the results of the test.

Part f (3 points)

Now use the `auto.arima()` function in the `forecast` library to fit an ARIMA model to this time series (put the entire tsibble into this function).

Feel free to check out this resource that discusses more about time series, including what the drift term is.

Part g (4 points)

Use the `forecast()` function from the `forecast` library to predict the future for 34 months. You should specify you are using the `forecast` library by typing `forecast::forecast()` since the `itsmr` library also has a `forecast()` function. The input of this function is your model you fit in part f. Use the model fit to the undifferenced data.

Then use the `autoplot()` function to plot the forecast.

Part h (8 points)

Plot the actual values for 1990 and 1991 along with the predicted values from the forecast and a shaded 95% interval bands. The `geom_ribbon()` function may be useful here. Is there any point where the true value is outside the interval? If so, report the year(s) and month(s) where that is the case.

Part i (5 points)

Use the `checkresiduals()` function in the `forecast` library to create a residual plot, an autocorrelation plot, and a histogram for the residuals. The input to this function should be your model fit with the `auto.arima()` function. Comment on whether the assumptions seem to be satisfied here. If you'd like, you can make a QQ-plot as well, but the histogram contains the same information.

Note: this function will also give you information from the Ljung-Box test, which tests if the residuals exhibit autocorrelation. A large p-value means we have no evidence of autocorrelation.

Problem 2 (66 points)

The National Institute of Diabetes and Digestive and Kidney Diseases conducted a study on 768 adult female Pima Native Americans living near Phoenix. The purpose of the study was to investigate factors related to diabetes. The response variable in this study (test) was whether or not the patient showed signs of diabetes (1 = yes, 0 = no). One explanatory variable considered was the plasma glucose concentration (in mg/dL) at 2 hours in an oral glucose tolerance test (glucose). Many other potential explanatory variables were considered as well. The goal here is to predict the diabetes status of the subject.

Part a (5 points)

Install and/or load the `faraway` package with the `pima` dataset. Set a seed and then split the data into a testing and a training set with the testing set containing 20% of the data values. When you do this, change the name of the variable "test" to "result" since it will be a bit confusing to call that test when we have a test dataset. Then make the `result` variable a factor.

Part b (4 points)

Fit a logistic regression model to the training set for predicting `result` from all other variables. Then obtain the predictions for the **test** set with a probability cutoff of 0.5 to predict if the person has diabetes. Create a confusion matrix using the `confusionMatrix()` function in the `caret` library. Print all of the output from that function that shows the confusion matrix and many associated statistics.

Part c (10 points)

Now let's use naïve Bayes to get predictions. First, find the predicted probability of being a 0 and a 1 for the first observation in the training set "by hand" like we did in the notes. To simplify this a bit, let's only use the variables `pregnant` and `bmi` here. Of course, use **R** to help you (the `dnorm()` function and some tidyverse functions will be especially nice), but don't use the `naive_bayes()` function yet. Note: these variables we are using to predict are numeric, so we will assume a normal distribution for both of them (probably not a good assumption, but let's go with it).

Part d (3 points)

Now use the `naive_bayes()` function in the `naivebayes` library to fit a model predicting result using `pregnant` and `bmi`. Find the predicted probabilities of being a 0 and a 1 for the first observation and show that it matches what you found in part c.

Part e (3 points)

Now fit a naïve Bayes model predicting `result` from **all** other variables. Then report the output of the `confusionMatrix()` function for this predictor when predicting on the **test** set.

Part f (3 points)

Now fit a naïve Bayes model predicting `result` from all other variables using a kernel instead of assuming a normal distribution. You can do this by setting `usekernel = TRUE` in the `naive_bayes()` function. Then report the output of the `confusionMatrix()` function for this predictor when predicting on the **test** set.

Part g (10 points)

Now let's use a support vector machine (SVM) for classifying diabetes using all other variables. We'll use the `svm()` function in the `e1071` library. Use a linear kernel, keep `scale = TRUE` (which is the default), and perform 5-fold cross validation to select the cost for costs between 0.1 and 5 with a step size of 0.1 (0.1, 0.2, 0.3, etc.). This will be similar to what you did back in Homework 3 problem 2h. Set a seed here. Like you did there, let's use Cohen's kappa to select the best cost value.

Part h (4 points)

Using the "best" cost value from part g, fit the SVM with a **linear** kernel and find the confusion matrix for predicting the **test** set.

Part i (10 points)

Now let's use a different kernel. Let's use the radial kernel for the SVM. This has two tuning parameters: the cost and γ. Let's use the `train()` function in the `caret` package to do 5-fold cross validation (set a seed, please) to select both of those parameters. However, the `train()` function uses σ instead of γ and C instead of cost.

Use the `trainControl()` function with the `trControl` option and the `tuneGrid` option in the `train()` function to perform this cross-validation. For `sigma`, use `seq(0.1, 2, by = 0.1)` and for `C` also use `seq(0.1, 2, by = 0.1)`. Put `method = "svmRadial"` in the `train()` function as well so it knows to use a SVM with the radial kernel. You will need to install the `kernlab` package for this to work.

By default, the "best" sigma and C values will be chosen with accuracy instead of Cohen's kappa. Make sure to choose the best values using kappa (although both metrics may choose the same "best" values).

Part j (4 points)

Fit the SVM with the **radial** kernel using the `svm()` function and the cross-validated cost and γ found in part i and print a confusion matrix for predicting on the **test** set.

Part k (10 points)

Compare the output of the `confusionMarix()` function for all of our classifiers: the logistic regression model, the naïve Bayes with a normal assumption, the naïve Bayes using kernel density estimation, the SVM with a linear kernel, and the SVM using a radial kernel. Which was best? Which was worst? How different were the results?