Predicting credit card defaults with logistic regression

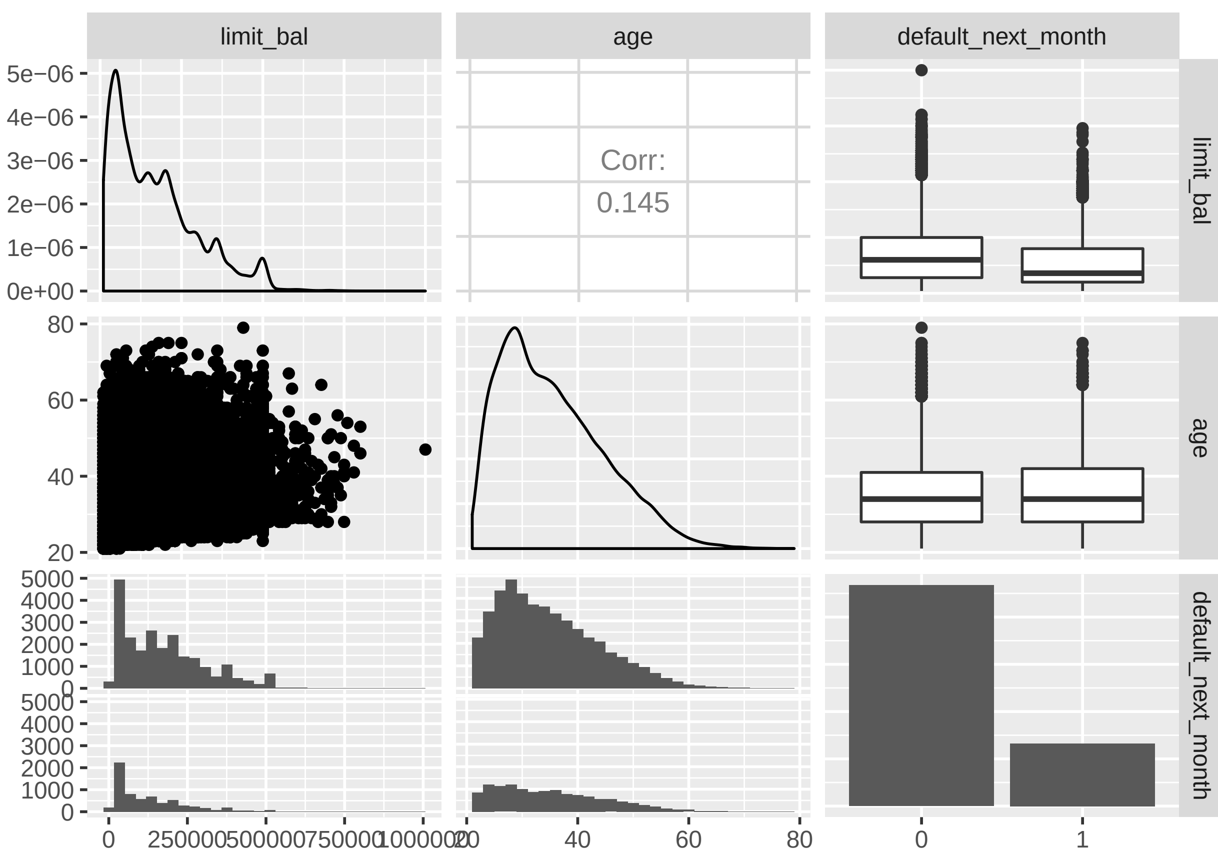
Instructions

Follow the steps below to familiarize yourself with the credit card default data set and prepare it for a machine learning analysis. You'll start by building a classifier using logistic regression and continue the model building in the next lesson.

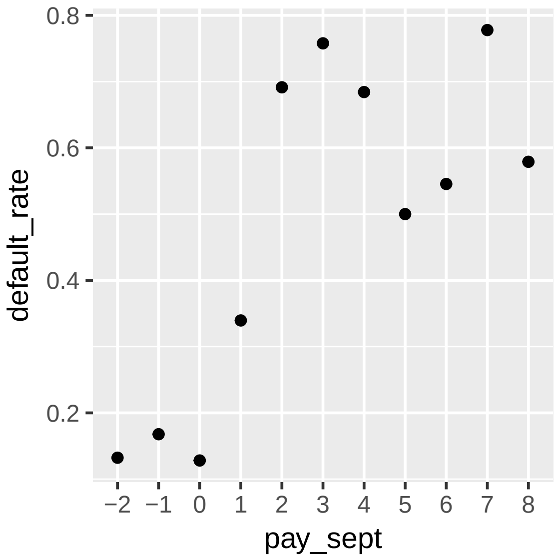
1. Read in the data using the `read.delim` function. Then use `ggplot2`, `ggpairs`, and `dplyr` to identify interesting relationships in the data. Write a short description of one interesting pattern you identified.

There is lots of exploratory data analysis to do. Below we show an example of exploring the numerical variables and then the pay_month variables.

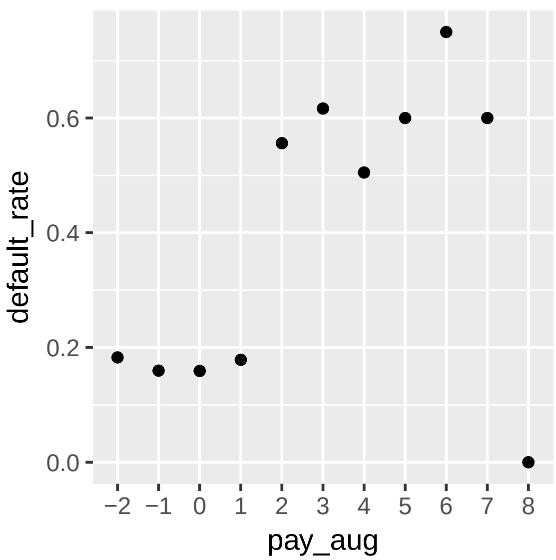
```
default$default_next_month = factor(default$default_next_month)

default$sex = factor(default$sex)
default$education = factor(default$education)
default$marriage = factor(default$marriage)
default$pay_sept = factor(default$pay_sept)
default$pay_aug = factor(default$pay_aug)
default$pay_july = factor(default$pay_july)
default$pay_june = factor(default$pay_june)
default$pay_may = factor(default$pay_may)
default$pay_april = factor(default$pay_april)

# example, look at numerical variables
ggpairs(select(default, limit_bal, age, default_next_month))
```



```r
# example, look at a few pay variables
sept_pay_default = default %>%
  group_by(pay_sept) %>%
  summarize( default_rate = sum(default_next_month == 1)/ n())

ggplot(data = sept_pay_default, aes(x = pay_sept, y =default_rate )) +
  geom_point()
```

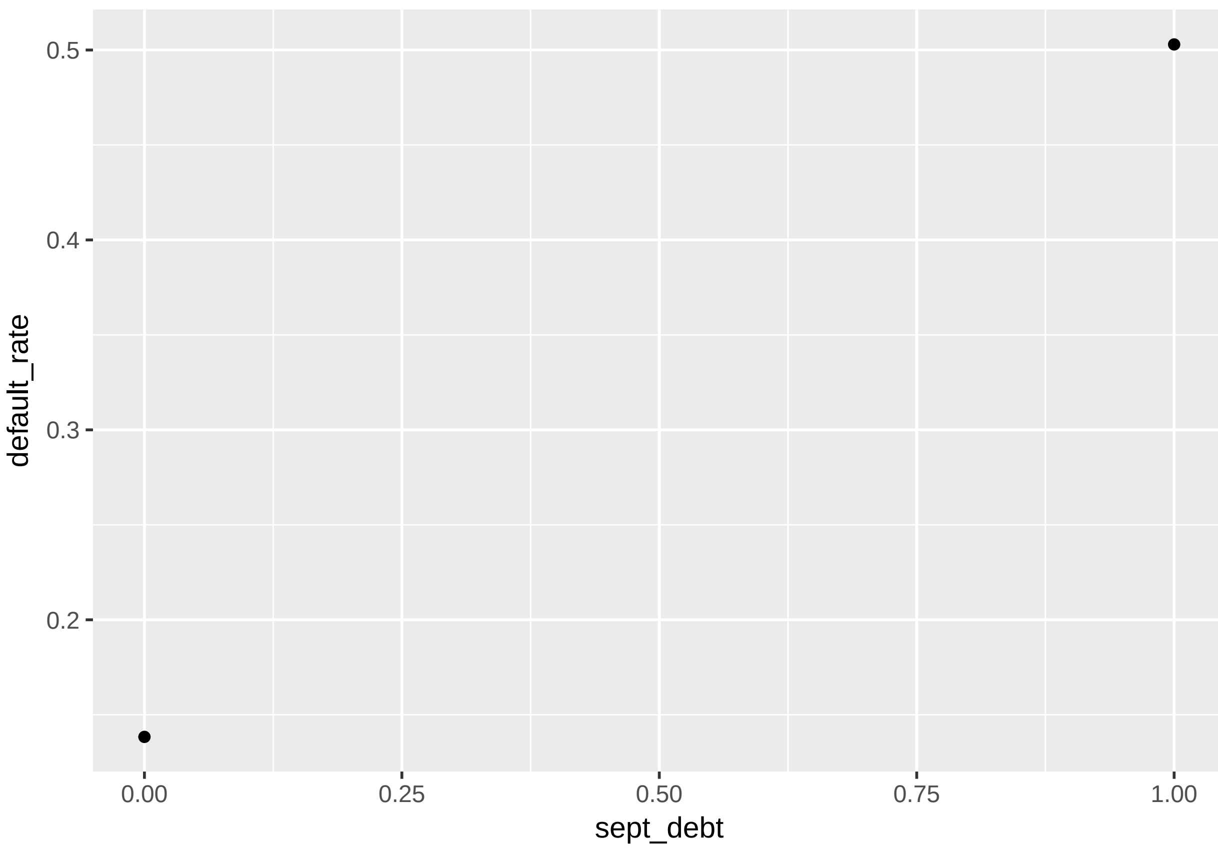



```r
aug_pay_default = default %>%
  group_by(pay_aug) %>%
  summarize( default_rate = sum(default_next_month == 1)/ n())

ggplot(data = aug_pay_default, aes(x = pay_aug, y =default_rate )) +
  geom_point()
```



2. Construct at least one new feature to include in model development. You might choose to create a new feature based on your findings from the exploratory data analysis. Plot or summarize the new variable and interpret the result. If your new variable is numerial, use color or a facet to show the relationship between your new feature and the outcome variable `default_next_month`.

```
default$sept_debt = ifelse(default$pay_sept %in% 1:8, 1, 0)

sept_debt_rate = default %>%
  group_by(sept_debt) %>%
  summarize( default_rate = sum(default_next_month == 1)/ n())

ggplot(sept_debt_rate, aes(x = sept_debt, y = default_rate)) +
  geom_point()
```



3. Use the `createDataPartition` function from the `caret` package to split the data into a training and testing set. Pre-process the data with `preProcess` as needed.

```
#use my sept_debt variable instead of pay_sept
default = select(default, -pay_sept)

in_train = createDataPartition(y = default$default_next_month, p = 0.8, list = FALSE)
default_train = default[in_train, ]
default_test = default[-in_train, ]

preprocess_steps = preProcess(default_train,
                              method = c("center", "scale", "nzv"))

default_train_proc = predict(preprocess_steps, default_train)
```

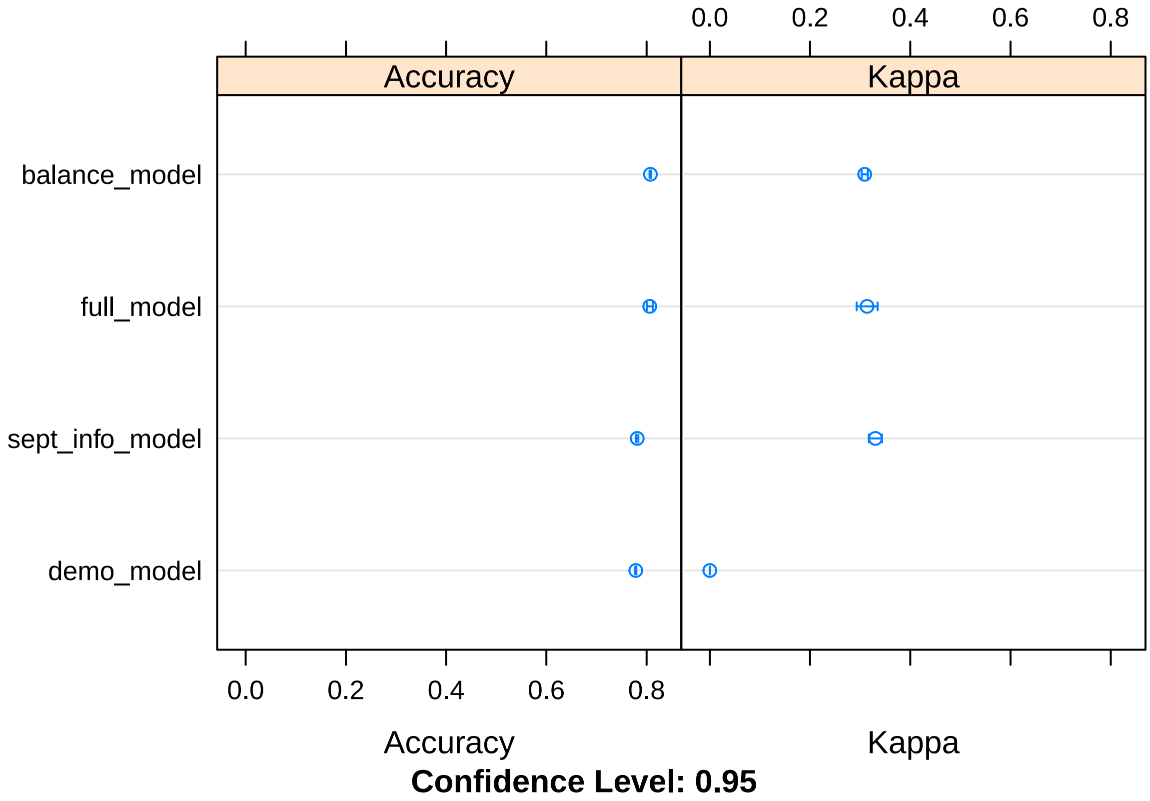
4. Fit at least 3 logistic regression models.

```
full_model = train(y = default_train$default_next_month,
                   x = select(default_train, -default_next_month),
                   method = "glm", family = "binomial")

demo_model = train(y = default_train$default_next_month,
                   x = select(default_train, age, education, marriage, sex),
                   method = "glm", family = "binomial")

balance_model = train(y = default_train$default_next_month,
                   x = select(default_train, starts_with("pay"), starts_with("bill"),
                              limit_bal, sept_debt),
                   method = "glm", family = "binomial")

sept_info_model = train(y = default_train$default_next_month,
                   x = select(default_train, contains("sept")),
                   method = "glm", family = "binomial")
```

5. Use the `dotplot` function to compare the accuracy of the models you constructed in 4. Which model performed the best in terms of predictive accuracy?

In terms of accuracy the full model performs the best. Interestingly, the kappa for the September balance information model is comparable or better than the other models. There were a lot of warnings running these models, it would be a good idea to further process the categorical variables. We could use `model.matrix` *to find near zero variance in the categories or do something similar to* `sept_debt` *for each month.*

```
results = resamples(list(full_model = full_model,
                         demo_model = demo_model,
                         balance_model = balance_model,
                         sept_info_model = sept_info_model))

dotplot(results)
```


0.0
0.2
0.4
0.6
0.8
Accuracy
Kappa
balance_model
full_model
sept_info_model
demo_model
0.0
0.2
0.4
0.6
0.8
Accuracy
Kappa
Confidence Level: 0.95