Credit Card Fraud

This dataset consists of credit card transactions in the western United States. It includes information about each transaction including customer details, the merchant and category of purchase, and whether or not the transaction was a fraud.

Note: You can access the data via the File menu or in the Context Panel at the top right of the screen next to Report, under Files. The data dictionary and filenames can be found at the bottom of this workbook.

Source: Kaggle The data was partially cleaned and adapted by DataCamp.

We've added some guiding questions for analyzing this exciting dataset! Feel free to make this workbook yours by adding and removing cells, or editing any of the existing cells.

Explore this dataset

Here are some ideas to get your started with your analysis

1. 🗺️ **Explore:** Find fraudulent cases per purchase category**
2. 📊 **Visualize:** Use a geospatial plot to visualize the fraud rates across different states.
3. 🔎 **Analyze:** Are older customers significantly more likely to be victims of credit card fraud?

🔍 Scenario: Accurately Predict Instances of Credit Card Fraud

This scenario helps you develop an end-to-end project for your portfolio.

Background: A new credit card company has just entered the market in the western United States. The company is promoting itself as one of the safest credit cards to use. They have hired you as their data scientist in charge of identifying instances of fraud. The executive who hired you has have provided you with data on credit card transactions, including whether or not each transaction was fraudulent.

Objective: The executive wants to know how accurately you can predict fraud using this data. She has stressed that the model should err on the side of caution: it is not a big problem to flag transactions as fraudulent when they aren't just to be safe. In your report, you will need to describe how well your model functions and how it adheres to these criteria.

You will need to prepare a report that is accessible to a broad audience. It will need to outline your motivation, analysis steps, findings, and conclusions.

You can query the pre-loaded CSV file using SQL directly. Here's a **sample query**, followed by some sample R code and outputs:

1. Explore : Find fraudulent cases per purchase category

- Answer: From the analysis conducted in R, it was found that the "Shopping Net" category has the highest fraud ratio. Specifically, a transaction in the "Shopping Net" category has an approximately 1.44% likelihood of being fraudulent.

```
## Find fraudulent cases per purchase category
suppressPackageStartupMessages(library(tidyverse))
library(ggplot2)
install.packages("viridis")
library(viridis)

ccf <- read_csv('credit_card_fraud.csv',show_col_types = FALSE) ## read csv
ccf_fraud <- ccf %>% filter(is_fraud==1) ## filter fraudumental transections
#1782
ccf_n <- nrow(ccf)

## grouping by categories

fraud_rates<- ccf %>%
                group_by(category) %>%
                summarise(total_trans = n(),
                            fraud_trans=sum(is_fraud==1,na.rm = TRUE),
                            fraud_per_rate= (fraud_trans/total_trans)*100) %>%
                            arrange(desc(fraud_per_rate))

print(fraud_rates)


# Bar plot of Fraud Rate by Category
ggplot(fraud_rates, aes(x = reorder(category, fraud_per_rate), y =
fraud_per_rate, fill = fraud_per_rate)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Fraud Rate by Purchase Category",
       x = "Category",
       y = "Fraud Rate (%)") +
  scale_fill_viridis(option = "C")

ggsave("plot_fraud_rate.png", width = 5, height = 5)
```

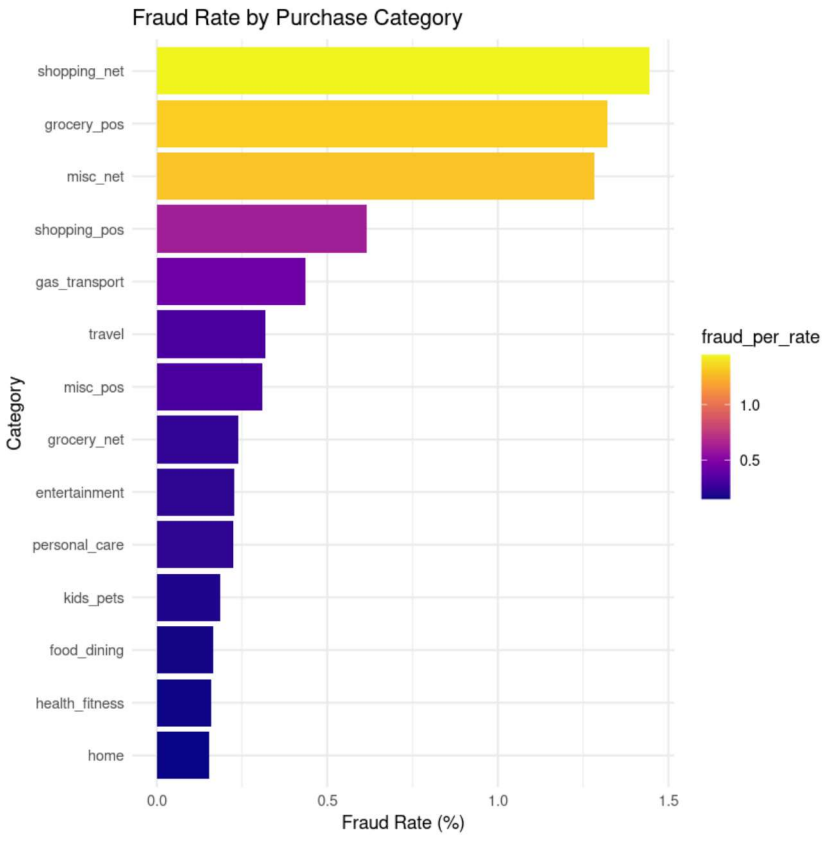
```
# Downloading packages -----------------------------------------------------
- Downloading viridis from CRAN ...             OK [2.9 Mb in 0.27s]
Successfully downloaded 1 package in 1.2 seconds.

The following package(s) will be installed:
- viridis [0.6.5]
These packages will be installed into "~/renv/library/linux-ubuntu-jammy/R-
4.4/x86_64-pc-linux-gnu".

# Installing packages ------------------------------------------------------
- Installing viridis ...                       OK [installed binary and cached
in 1.3s]
Successfully installed 1 package in 1.4 seconds.
```

```
Loading required package: viridisLite
```

```
# A tibble: 14 × 4
   category       total_trans fraud_trans fraud_per_rate
   <chr>                <int>       <int>          <dbl>
 1 shopping_net         26379         381          1.44
 2 grocery_pos          32732         433          1.32
 3 misc_net             16898         217          1.28
 4 shopping_pos         30329         187          0.617
 5 gas_transport        35089         153          0.436
 6 travel               10322          33          0.320
 7 misc_pos             20024          62          0.310
 8 grocery_net          11355          27          0.238
 9 entertainment        24222          55          0.227
10 personal_care        24406          55          0.225
11 kids_pets            29704          55          0.185
12 food_dining          23038          38          0.165
13 health_fitness       22593          36          0.159
14 home                 32516          50          0.154
```



📊 Visualize: Use a geospatial plot to visualize the fraud rates across different states.

Data Dictionary

transdatetrans_time	Transaction DateTime
merchant	Merchant Name
category	Category of Merchant
amt	Amount of Transaction
city	City of Credit Card Holder
state	State of Credit Card Holder
lat	Latitude Location of Purchase
long	Longitude Location of Purchase
city_pop	Credit Card Holder's City Population
job	Job of Credit Card Holder
dob	Date of Birth of Credit Card Holder
trans_num	Transaction Number
merch_lat	Latitude Location of Merchant
merch_long	Longitude Location of Merchant
is_fraud	Whether Transaction is Fraud (1) or Not (0)

Source of dataset. The data was partially cleaned and adapted by DataCamp.

Don't know where to start?

Challenges are brief tasks designed to help you practice specific skills:

- 🗺️ **Explore**: What types of purchases are most likely to be instances of fraud? Consider both product category and the amount of the transaction.
- 📊 **Visualize**: Use a geospatial plot to visualize the fraud rates across different states.
- 🔎 **Analyze**: Are older customers significantly more likely to be victims of credit card fraud?

Scenarios are broader questions to help you develop an end-to-end project for your portfolio:

A new credit card company has just entered the market in the western United States. The company is promoting itself as one of the safest credit cards to use. They have hired you as their data scientist in charge of identifying instances of fraud. The executive who hired you has have provided you with data on credit card transactions, including whether or not each transaction was fraudulent.

The executive wants to know how accurately you can predict fraud using this data. She has stressed that the model should err on the side of caution: it is not a big problem to flag transactions as fraudulent when they aren't just to be safe. In your report, you will need to describe how well your model functions and how it adheres to these criteria.

You will need to prepare a report that is accessible to a broad audience. It will need to outline your motivation, analysis steps, findings, and conclusions.

🔎 Analyze: trending model for detected fraudulent?

My process for analysing

**1.Clean data **: check missing value is none

2.Handling Imbalanced Data: The dataset was highly imbalanced, with a fraud-to-non-fraud ratio of 0.00527:1. To address this, I applied upsampling to balance the dataset.

3.Convert varible for using trend model in valid format type

4.Using logittic regression for trend datadet in binary class using variable amt+time_to_second for train model in train dataset

5.predicting accuracy in train and test dataset ,both gives about 85% ,that can definded the model did not overfited with only train data

6.using confusion matrix find precision ,recall and F1 score for making that model has accurate for dectecing fraudulent excextly.

Conclusion : From test data set rusult

Precision : The high precision indicates that when the model predicts fraud, it's very likely to be correct, which is crucial for minimizing false positives (non-fraudulent transactions being flagged as fraud). 0.9336055

recall: The recall is somewhat lower, indicating that the model might be missing a portion of the fraudulent transactions (false negatives). 0.7468549

F1 score : The F1 score indicates a good overall performance, particularly balancing the need for precision and recall (A model with an F1 score close to 1 means it has a high balance between precision and recall, indicating strong overall predictive performance.) 0.8298533

```
# Load required libraries
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(dplyr))
install.packages("caret")
suppressPackageStartupMessages(library(caret))
install.packages("mlbench")
library(mlbench)
# Load the dataset

ccf <- read_csv('credit_card_fraud.csv', show_col_types = FALSE)

# Check for missing values
sum(is.na(ccf$is_fraud))  # Count total Null values

# Check class distribution
table(ccf$is_fraud)

# Calculate fraud ratio
is_frauds <- sum(ccf$is_fraud == 1)
not_frauds <- sum(ccf$is_fraud == 0)
ratio <- is_frauds / not_frauds
print(ratio)  # Check fraud-to-non-fraud ratio

# Apply upsampling
set.seed(42)
ccf_over <- upSample(x = ccf %>% select(-is_fraud), y = as.factor(ccf$is_fraud))
nrow(ccf_over)

# rename column Class to is_fraud
ccf_over <- ccf_over %>%
  rename(is_fraud = Class)

ccf_over$is_fraud <- factor(ccf_over$is_fraud, levels = c(0, 1), labels = c("No",
"Yes"))

table(ccf_over$is_fraud)  # Should return counts for "No" and "Yes"

# Convert to Date time format
ccf_over$dob <- as.Date(ccf_over$dob, format = "%Y-%m-%d")

# creat age colums in data set
ccf_over$age <- as.numeric(difftime(Sys.Date(), ccf_over$dob, units = "days")) /
365

ccf_over$trans_date_trans_time <- as.POSIXct(ccf_over$trans_date_trans_time,
                                             format="%Y-%m-%d %H:%M:%S")

# Convert HH:MM:SS into total seconds
ccf_over$time_in_seconds <- as.numeric(format(ccf_over$trans_date_trans_time,
"%H")) * 3600 +
```

```
                              as.numeric(format(ccf_over$trans_date_trans_time,
"%M")) * 60 +
                              as.numeric(format(ccf_over$trans_date_trans_time,
"%S"))

# convert string dataset to factors
ccf_over$merchant <- as.factor(ccf_over$merchant)
ccf_over$category <- as.factor(ccf_over$category)
ccf_over$city <- as.factor(ccf_over$city)
ccf_over$state <- as.factor(ccf_over$state)
ccf_over$job <- as.factor(ccf_over$job)


# split data
split_data <- function(data) {
  set.seed(42)
  n <- nrow(data)
  id <- sample(1:n, size=0.7*n)
  train_df <- data[id,]
  test_df <- data[-id,]
  return( list(train_ccf=train_df,test_ccf=test_df))
}
prep_ccf <- split_data(ccf_over)
sum(is.na(prep_ccf))
table(prep_ccf$train_ccf$is_fraud)
table(prep_ccf$test_ccf$is_fraud)


## logistic regression method = "glm"
set.seed(42)
## repeated k-fold cv
ctrl <- trainControl(method = "cv",
                     number = 5) # k

# Train Logistic Regression
logit_model <- train(is_fraud ~ amt+time_in_seconds,
                     data = prep_ccf$train_ccf,
                     method = "glm",
                     family = binomial,
                     metric = "Accuracy",
                     trControl = ctrl)

# Check Model
summary(logit_model)
## final model
logit_model$finalModel

# Generate Predictions train dataset
predict_train <- predict(logit_model,prep_ccf$train_ccf)

# Create the confusion matrix
```

```
conf_matrix_train <- confusionMatrix(factor(predict_train),
factor(prep_ccf$train_ccf$is_fraud))

# Print the confusion matrix
cat("Confusion Matrix for Train Data:\n")
print(conf_matrix_train)

# Generate Predictions test dataset
predict_test <- predict(logit_model, prep_ccf$test_ccf)

# Get predicted probabilities for the test dataset
pred_prob_test <- predict(logit_model, prep_ccf$test_ccf, type = "prob")

# Apply the threshold to classify transactions as fraud if probability >= 0.5
predicted_fraud_test <- ifelse(pred_prob_test$Yes >= 0.5 ,"Yes","No")

# Create the confusion matrix
conf_matrix_test <- confusionMatrix(factor(predicted_fraud_test),
factor(prep_ccf$test_ccf$is_fraud))

# Print the confusion matrix
cat("Confusion Matrix for test Data:\n")
print(conf_matrix_test)


# Convert confusion matrix from list to matrix
conf_matrix_matrix <- conf_matrix_test$table

# Print the matrix
print(conf_matrix_matrix)

precision <-
conf_matrix_matrix[2,2]/(conf_matrix_matrix[2,1]+conf_matrix_matrix[2,2])
cat("Precision : \n",precision,"\n")

recall <-
conf_matrix_matrix[2,2]/(conf_matrix_matrix[1,2]+conf_matrix_matrix[2,2])
cat("recall: \n",recall,"\n")

F1_score <- 2*(precision*recall)/(precision+recall)
cat("F1 score : \n",F1_score)
```

```
# Downloading packages -----------------------------------------------------
- Downloading caret from CRAN ...               OK [3.4 Mb in 0.25s]
Successfully downloaded 1 package in 1.1 seconds.

The following package(s) will be installed:
- caret [7.0-1]
These packages will be installed into "~/renv/library/linux-ubuntu-jammy/R-
4.4/x86_64-pc-linux-gnu".

# Installing packages ------------------------------------------------------
```

```
- Installing caret ...                          OK [installed binary and
cached in 2.8s]
Successfully installed 1 package in 2.8 seconds.
# Downloading packages -----------------------------------------------------
- Downloading mlbench from CRAN ...             OK [1 Mb in 0.33s]
Successfully downloaded 1 package in 0.43 seconds.

The following package(s) will be installed:
- mlbench [2.1-6]
These packages will be installed into "~/renv/library/linux-ubuntu-jammy/R-
4.4/x86_64-pc-linux-gnu".

# Installing packages
```

```
0
```

```
     0      1
337825   1782
```

```
[1] 0.00527492
```

```
675650
```

```
    No    Yes
337825 337825
```

```
0
```

```
    No    Yes
236158 236796
```

```
    No    Yes
101667 101029
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

```

Call:
NULL

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.288e+00  7.928e-03 -162.42   <2e-16 ***
amt              7.634e-03  2.794e-05  273.21   <2e-16 ***
time_in_seconds -4.113e-06  1.320e-07  -31.17   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 655653  on 472953  degrees of freedom
Residual deviance: 408576  on 472951  degrees of freedom
AIC: 408582

Number of Fisher Scoring iterations: 7
```

```

Call:  NULL

Coefficients:
    (Intercept)              amt  time_in_seconds
     -1.288e+00        7.634e-03       -4.113e-06

Degrees of Freedom: 472953 Total (i.e. Null);  472951 Residual
Null Deviance:      655700
Residual Deviance: 408600   AIC: 408600
```

```
Confusion Matrix for Train Data:
Confusion Matrix and Statistics

          Reference
Prediction     No    Yes
       No  223643  59073
       Yes  12515 177723

               Accuracy : 0.8486
                 95% CI : (0.8476, 0.8497)
    No Information Rate : 0.5007
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6974

 Mcnemar's Test P-Value : < 2.2e-16
```

```
            Sensitivity : 0.9470
            Specificity : 0.7505
         Pos Pred Value : 0.7911
         Neg Pred Value : 0.9342
             Prevalence : 0.4993
```

Write R code or tell our AI what to do