Code

R Notebook

Jenna Leali - Project 3 - Oct 18

Load Libraries

Hide

```
library(tidyverse)
library(ggplot2)
library(readr)
library(summarytools)
```

Read in CSV Files

Hide

```
household_data <- read.csv("/Users/jenna/Desktop/household_energy_usage_regression.csv")
```

Hide

```
card_data <- read.csv("/Users/jenna/Desktop/creditcard.csv")
```

Part 1: Linear Regression with Household Energy Consumption Data

Prepare the data

Reading the Data

Hide

```
head(household_data)
```

- From examining the first few rows of the dataset, I noticed that it starts with records from September 26, 2019, and provides hourly energy consumption readings along with various weather-related variables.
- The datetime column, which contains date and time information, will be essential for time-based analysis, and I'll need to consider how to utilize this information effectively for modeling.
- I observed that the dataset consists of multiple numeric columns, including temperature measurements

in both Fahrenheit and Celsius, humidity, and energy consumption in kilowatt-hours.

Summary Stats

Hide

```
descr(household_data)
```

- I observed that the mean energy consumption is approximately 1.97 kilowatt-hours, with a standard deviation of about 1.12 kWh. The data ranges from a minimum of 0.29 kWh to a maximum of 7.64 kWh.
- The summary statistics also reveal insights into the weather-related variables. For example, the temperatureC column has a mean of 25.44 degrees Celsius, while the temperatureF column shows an average of 77.79 degrees Fahrenheit. The temperature values exhibit relatively low variation, as indicated by the low coefficients of variation.
- Additionally, examining skewness and kurtosis values, I noted that energy consumption follows a slightly positively skewed distribution (skewness ≈ 0.66), suggesting that most values are concentrated around the mean. The kurtosis value, around 0.32, indicates that the distribution has moderate tails.

Data Structure

Hide

```
str(household_data)
```

- str reveals that the dataset consists of 10,055 observations and 6 variables. This indicates a substantial amount of data to work with for modeling energy consumption.
- I observed that the datetime column is of character (chr) data type, which is expected for timestamps.
- The other five variables, including kwh, temperatureF, temperatureC, humidity, and dewpointC, are all of numeric (num) data type. This aligns with expectations, as these variables represent numerical measurements related to energy consumption and weather conditions.

Column Names

Hide

```
colnames(household_data)
```

Data Cleaning

Missing Values

Hide

```
colSums(is.na(household_data))
```

Hide

```
household_data <- na.omit(household_data)
```

- Given the relatively large size of my dataset, which contains thousands of observations, I made the decision to remove the rows with missing values in the columns temperatureF, temperatureC, humidity, and dewpointC. This approach allows me to retain the majority of the data while effectively addressing the missing data issue.

Outliers in Dataset

Hide

```
boxplot(household_data$kwh, main="kwh Box Plot", ylab="kwh")
boxplot(household_data$temperatureF, main="temperatureF Box Plot", ylab="temperature
F")
boxplot(household_data$temperatureC, main="temperatureC Box Plot", ylab="temperature
C")
boxplot(household_data$humidity, main="humidity Box Plot", ylab="humidity")
boxplot(household_data$dewpointC, main="dewpointC Box Plot", ylab="dewpointC")
```

- I've chosen to keep outliers in the numeric categories of the dataset, such as energy consumption, temperature, and humidity. This decision is primarily driven by the presence of valid data points representing extremely high energy consumption during events such as heatwaves, which reflect real-world scenarios.
- Retaining outliers is beneficial because it enhances the model's ability to generalize to extreme scenarios.

Correlation

Hide

```
correlations <- cor(household_data[, c("kwh", "temperatureF", "humidity", "dewpoint
C")])
print(correlations)
```

- I observed a strong positive correlation of approximately 0.636 between energy consumption and temperature in Fahrenheit. This means that as the temperature in Fahrenheit increases, I noticed a tendency for energy consumption to rise as well. It indicates that higher temperatures are linked to increased energy usage.
- I also identified a negative correlation of around -0.415 between energy consumption and humidity. This suggests that higher humidity levels were associated with reduced energy consumption. As humidity increased, I noticed a tendency for energy usage to decrease.
- Furthermore, there was a substantial positive correlation of approximately 0.635 between energy consumption and the dew point in Celsius. Similar to the temperature variable, I found that higher dew point values were linked to increased energy consumption. This relationship highlights the impact of dew point on energy usage in my dataset.

- The correlation between energy consumption and itself is, of course, perfect and equal to 1. This self-correlation signifies a perfect positive relationship, which is expected as energy consumption correlates perfectly with itself.
- The positive correlation between temperature in Fahrenheit and dew point in Celsius (approximately 0.9999821) is almost perfect. This near-perfect correlation suggests that these two weather-related variables are highly linearly related.
- The correlations mentioned above are all relatively strong. This implies that the weather-related variables, temperature, humidity, and dew point, have a notable influence on household energy consumption in the dataset.

Create new Variable

Hide

```
household_data <- household_data
household_data$month <- as.integer(format(as.Date(household_data$datetime), "%m"))
household_data$day <- as.integer(format(as.Date(household_data$datetime), "%d"))

household_data$season <- ifelse(household_data$month %in% c(3, 4, 5), "Spring",
                                ifelse(household_data$month %in% c(6, 7, 8), "Summe
r",
                                ifelse(household_data$month %in% c(9, 10, 11), "Fal
l", "Winter")))
```

Hide

```
head(household_data)
```

- I decided to create the season variable because it allows me to capture the potential influence of seasonal changes on household energy consumption.
- This new variable will be beneficial to the model as it provides a structured way to incorporate seasonal patterns into the analysis. By including ‘season’ as a predictor, the model can account for variations in energy consumption across different times of the year, improving its ability to make accurate predictions and enhancing its overall performance.

Building the Model

Data Splitting

Hide

```
library(caret)
set.seed(123)
trainIndex <- createDataPartition(household_data$kwh, p = 0.8, list = FALSE)
trainData <- household_data[trainIndex, ]
testData <- household_data[-trainIndex, ]
```

Build Linear Regression Model

Hide

```
lm_model <- lm(kwh ~ temperatureF + humidity + dewpointC + season, data = trainData)
```

Summary of Model

Hide

```
summary(lm_model)
```

Check for correlations between ‘dewpointC’ and other predictors

Hide

```
cor(trainData$dewpointC, trainData$temperatureF)
cor(trainData$dewpointC, trainData$humidity)
```

- I noticed that there is an extremely high correlation (almost 1) between the variables “temperatureF” and “dewpointC” in the dataset. This high correlation indicates that these two variables move almost identically, which can cause issues in the linear regression model.
- I decided it is best to try and build the model after removing “dewpointC” from the model.

Re-Building the Model

Hide

```
lm_model_no_dewpoint <- lm(kwh ~ temperatureF + humidity + season, data = trainData)
```

Summary of Model

Hide

```
summary(lm_model_no_dewpoint)
```

Interpret the Models Fit

- After removing dewpointC, I observed that the model’s performance and key statistics remained very similar to the previous model with dewpointC. This suggests that dewpointC may not have been a significant predictor for energy consumption in this context, and its removal didn’t have a substantial

impact on the model's predictive power.

- The coefficient for temperatureF is positive (0.0707), indicating that an increase in temperature is associated with a significant increase in household energy consumption. For every one-unit rise in temperature (in Fahrenheit), energy consumption is expected to increase by approximately 0.0707 units.
- The coefficient for humidity is negative (-2.1061), signifying that an increase in humidity is linked to a substantial decrease in energy consumption. Specifically, for every one-unit rise in humidity, energy consumption is expected to decrease by approximately 2.1061 units.
- The season variables reveal seasonal effects on energy consumption. Winter exhibits the most significant impact, leading to an expected decrease of approximately 0.3994 units in energy consumption.
- The multiple R-squared value of 0.4647 suggests that approximately 46.47% of the variance in energy consumption is explained by the included predictors. This indicates that the model has a moderate level of explanatory power regarding energy consumption.
- The high F-statistic with a very low p-value (p-value: < 2.2e-16) indicates that the model as a whole is statistically significant.
- In my analysis, it's evident that the temperatureF variable has the most substantial role in predicting household energy consumption (kwh). An increase in temperature, as measured in Fahrenheit, is strongly associated with a significant rise in energy consumption. This suggests that temperature fluctuations play a crucial role in influencing energy usage patterns.

Evaluate the model's performance

Hide

```
predicted_values <- predict(lm_model_no_dewpoint, newdata = trainData)

mae <- mean(abs(trainData$kwh - predicted_values))

mse <- mean((trainData$kwh - predicted_values)^2)

rmse <- sqrt(mse)

ss_residual <- sum((trainData$kwh - predicted_values)^2)
ss_total <- sum((trainData$kwh - mean(trainData$kwh))^2)
rsquared <- 1 - (ss_residual / ss_total)

cat("Mean Absolute Error (MAE):", mae, "\n")
cat("Mean Squared Error (MSE):", mse, "\n")
cat("Root Mean Squared Error (RMSE):", rmse, "\n")
cat("R-squared (R²):", rsquared, "\n")
```

- My model achieved a MAE of 0.6137, which means, on average, it predicted household energy consumption within approximately 0.6137 units of the actual values.
- The model succeeded in explaining a significant portion of the variance in energy consumption, with an R-squared (R^2) value of 0.4647. This means that approximately 46.47% of the variability in energy usage

was accounted for by the included predictor variables.

Part 2: Binary Classification

Prepare the data

Examine the Structure

Hide

```
str(card_data)
```

- Dataset Size: I noticed that the dataset is fairly large, with 284,807 observations.
- Feature Set: In examining the dataset, I found that it contains a total of 31 variables. These variables include “Time,” “Amount,” and 28 features labeled as “V1” through “V28.”
- Binary Class: It’s worth noting that the target variable, “Class,” is binary. Specifically, a value of 1 is used to indicate a fraudulent transaction, while a value of 0 represents a non-fraudulent transaction.

Show the first few rows of the dataset

Hide

```
head(card_data)
```

Check for missing values

Hide

```
any(is.na(card_data))
```

- There are no missing values in the dataset

Check for class imbalance

Hide

```
barplot(table(card_data$Class), main = "Class Distribution", xlab = "Class (0: Non-Fr
aud, 1: Fraud)")
```

Hide

```
table(card_data$Class)
```

- Class 0: Non-fraudulent transactions, with 284,315 instances.
- Class 1: Fraudulent transactions, with only 492 instances.
- When building the model, it might be biased toward predicting the majority class (Class 0) due to the

substantial class size difference.

Hide

```
summary(card_data$Time)
summary(card_data$Amount)
```

- Time Distribution: I observed that the Time variable, representing the time interval between transactions, has a wide range of values. The distribution ranges from a minimum of 0 seconds to a maximum of 172,792 seconds, indicating that the transactions in the dataset span a significant period of time. The majority of transactions fall within the range of 54,202 to 139,320 seconds, with a median value of 84,692 seconds.
- Amount Distribution: I also noticed that the Amount variable, which represents the transaction amounts, exhibits a positively skewed distribution. The majority of transaction amounts are relatively small, with a mean of 88.35 and a median of 22.00. However, there are some high-value transactions, as indicated by the maximum value of 25,691.16.

Time series graph for time variable

Hide

```
fraud_cases <- card_data$Time[card_data$Class == 1]
plot(fraud_cases, type = "l", xlab = "Time", ylab = "Number of Fraud Cases")
```

- The line on the graph is moving upward. This upward trend suggests that, over time, there has been an increase in the number of detected credit card fraud cases. In other words, as time progresses, more fraud cases appear to be occurring or getting detected.

Boxplot for amount variable

Hide

```
boxplot(Amount ~ Class, data = card_data, xlab = "Class", ylab = "Transaction Amoun
t")
```

- Boxplot for Class 0 (Non-fraudulent transactions): The boxplot appears as a line around 0, which suggests that the transaction amounts for non-fraudulent transactions are concentrated around the lower values. There are numerous dots going upward, indicating that there are some transactions with higher amounts but relatively few in number. The presence of more dots for 0 suggests that there are many non-fraudulent transactions with lower amounts.
- Boxplot for Class 1 (Fraudulent transactions): The boxplot for Class 1 also appears as a line around 0, suggesting that the transaction amounts for fraudulent transactions are mainly concentrated around the lower values. There are numerous dots going upward, indicating that some fraudulent transactions have higher amounts, but they are still relatively few in number. The presence of fewer dots for 1 suggests that there are fewer fraudulent transactions with higher amounts compared to non-fraudulent transactions.

Min-Max Scaling for amount variable

Hide

```
min_amount <- min(card_data$Amount)
max_amount <- max(card_data$Amount)

card_data$Amount <- (card_data$Amount - min_amount) / (max_amount - min_amount)
```

Building logistic regression classification model

Split the data

Hide

```
set.seed(123)
train_indices <- sample(1:nrow(card_data), 0.7 * nrow(card_data))
train_data <- card_data[train_indices, ]
test_data <- card_data[-train_indices, ]

mean_train <- mean(train_data$Amount)
sd_train <- sd(train_data$Amount)

train_data$Amount <- (train_data$Amount - mean_train) / sd_train
test_data$Amount <- (test_data$Amount - mean_train) / sd_train
```

Handling errors

Hide

```
train_data$Class <- as.factor(train_data$Class)
test_data$Class <- as.factor(test_data$Class)
```

Hide

```
levels(train_data$Class)
levels(test_data$Class)
```

Hide

```
predicted_class <- factor(predicted_class, levels = levels(test_data$Class))

confusion_matrix <- confusionMatrix(predicted_class, test_data$Class)
```

Hide

```
levels(predicted_class)
levels(test_data$Class)
anyNA(predicted_class)
anyNA(test_data$Class)
```

Hide

```
test_data$Class <- factor(test_data$Class, levels = c("0", "1"))

threshold <- 0.5
predicted_probabilities <- predict(model, newdata = test_data, type = "response")
predicted_class <- ifelse(predicted_probabilities >= threshold, "1", "0")
predicted_class <- factor(predicted_class, levels = c("0", "1"))
```

- I tried to ensure that the “Class” variable was recognized as a factor with the correct levels to address the “Error: data and reference should be factors with the same levels” issue. Frustrated with the recurring error, I decided to manually reconstructe the problematic variables involved in the error.

Hide

```
library(caret)

set.seed(123)

train_data$Class <- as.factor(train_data$Class)
test_data$Class <- as.factor(test_data$Class)

model <- glm(Class ~ ., data = train_data, family = binomial)

predictions <- predict(model, newdata = test_data, type = "response")

threshold <- 0.5
predicted_class <- ifelse(predictions >= threshold, 1, 0)

conf_matrix <- table(Predicted = predicted_class, Actual = test_data$Class)

TP <- conf_matrix["1", "1"]
TN <- conf_matrix["0", "0"]
FP <- conf_matrix["1", "0"]
FN <- conf_matrix["0", "1"]

accuracy <- (TP + TN) / sum(conf_matrix)

precision <- TP / (TP + FP)

recall <- TP / (TP + FN)

f1_score <- 2 * (precision * recall) / (precision + recall)

conf_matrix
cat("Accuracy: ", accuracy, "\n")
cat("Precision: ", precision, "\n")
cat("Recall (Sensitivity): ", recall, "\n")
cat("F1-Score: ", f1_score, "\n")
```

- Accuracy: I observed that the model demonstrates a high level of overall accuracy, indicating that it correctly classifies the majority of cases. However, I am aware that in imbalanced datasets like this one, high accuracy can be misleading due to the dominance of non-fraudulent transactions (class 0).
- Precision: The precision of approximately 0.87 impressed me. It signifies that when the model predicts a transaction as fraudulent (class 1), it is correct about 87% of the time.
- Recall (Sensitivity): The model finds about 61% of the real fraud cases.
- F1-Score: The F1-Score is like a balanced measure. It considers both how often the model is right about fraud and how often it finds fraud cases. In this case, it's around 0.72, which is a pretty good balance.

Building the model while attempting to account for the imbalance

Hide

```
set.seed(123)

train_data$Class <- as.factor(train_data$Class)
test_data$Class <- as.factor(test_data$Class)

class_weights <- ifelse(train_data$Class == 1, 10, 1)

model <- glm(Class ~ ., data = train_data, family = binomial, weights = class_weight
s)

predictions <- predict(model, newdata = test_data, type = "response")

threshold <- 0.5
predicted_class <- ifelse(predictions >= threshold, "1", "0")

conf_matrix <- table(Predicted = predicted_class, Actual = test_data$Class)
TP <- conf_matrix["1", "1"]
TN <- conf_matrix["0", "0"]
FP <- conf_matrix["1", "0"]
FN <- conf_matrix["0", "1"]

accuracy <- (TP + TN) / sum(conf_matrix)
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
f1_score <- 2 * (precision * recall) / (precision + recall)

conf_matrix
cat("Accuracy: ", accuracy, "\n")
cat("Precision: ", precision, "\n")
cat("Recall (Sensitivity): ", recall, "\n")
cat("F1-Score: ", f1_score, "\n")
```

- Accuracy: The model is still doing well in getting most predictions right. But because there are many more non-fraudulent cases, it's not too surprising that accuracy is high.
- Precision: The precision, which used to be very high, has decreased a bit to around 76%. This means that when the model says a transaction is fraud, it's right about 76% of the time. It's still pretty accurate.
- Recall (Sensitivity): The recall has gotten better and is now around 81%. This means the model is capturing about 81% of the real fraudulent transactions. That's a big improvement from before.
- F1-Score: The F1-Score, which is about 0.79, shows a good balance between precision and recall.

Evalute the Model

Hide

```
conf_matrix <- table(Predicted = predicted_class, Actual = test_data$Class)

TP <- conf_matrix["1", "1"]
TN <- conf_matrix["0", "0"]
FP <- conf_matrix["1", "0"]
FN <- conf_matrix["0", "1"]

accuracy <- (TP + TN) / sum(conf_matrix)

precision <- TP / (TP + FP)

recall <- TP / (TP + FN)

f1_score <- 2 * (precision * recall) / (precision + recall)

library(pROC)

roc_auc <- auc(roc(test_data$Class, predicted_probabilities))

precision_values <- c(1, precision, 0)
recall_values <- c(0, recall, 1)
pr_auc <- sum(diff(recall_values) * precision_values[-length(precision_values)])

cat("Confusion Matrix:\n")
print(conf_matrix)
cat("\nAccuracy: ", accuracy, "\n")
cat("Precision: ", precision, "\n")
cat("Recall (Sensitivity): ", recall, "\n")
cat("F1-Score: ", f1_score, "\n")
cat("ROC-AUC: ", roc_auc, "\n")
cat("PR-AUC: ", pr_auc, "\n")
```

- The model achieved a high level of accuracy, with an accuracy score of 0.9992. This means it correctly classifies the majority of transactions.
- The precision of 0.7622 is a notable metric, indicating that when the model predicts a transaction as fraudulent, it is correct approximately 76.22% of the time.
- The recall, or sensitivity, score of 0.8117 shows that the model successfully captures about 81.17% of the actual fraudulent transactions.
- The F1-Score, which balances precision and recall, is at 0.7862. This suggests that the model maintains a good trade-off between accurately identifying fraud and minimizing false positives.
- The ROC-AUC score of 0.9740 demonstrates the model's strong ability to differentiate between the two classes, emphasizing its effectiveness in classification
- Overall, these results suggest that the logistic regression model is performing well in identifying fraudulent transactions in an imbalanced dataset. The combination of high precision, recall, and AUC scores signifies its potential for practical use in real-world fraud detection scenarios.