Statistical Learning Assignment01

Jinyi Tzeng 47969938

Introduction

Identifying the factors associated with insurance claim expenses is crucial for pricing and risk evaluation in the insurance industry. The automobile insurance dataset used in this analysis is zero-inflated, but has been preprocessed to focus on positive claim amounts.

This report aims to model the non-zero claims using various machine learning techniques, with the goal of finding the **best-performing model** based on root mean squared error (RMSE).

Part (a): Data Loading

1. Load the dataset autoinsurance.csv.

2. Examine its structure and identify variable types.

```
skimr::skim(autoinsurance)
```

Table 1: Data summary

Name	autoinsurance
Number of rows	20000
Number of columns	11
_____	
Column type frequency:	
character	2
numeric	9
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
geo_area	0	1	5	8	0	3	0
vehicle_type	0	1	3	7	0	4	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	42.03	16.48	16	29.00	41.0	53.0	89.0	
est_value	0	1	40729.48	20141.99	5000	28714.75	34473.54	45361.5	125000.0	
miles_driven	0	1	13954.95	3806.18	2545	11430.00	13547.51	15998.04	41124.0	
college_grad_ind	0	1	0.26	0.44	0	0.00	0.0	1.0	1.0	
speeding_ticket_ind	0	1	0.15	0.35	0	0.00	0.0	0.0	1.0	
hard_braking_ind	0	1	0.34	0.47	0	0.00	0.0	1.0	1.0	
late_driving_ind	0	1	0.05	0.22	0	0.00	0.0	0.0	1.0	
clean_driving_ind	0	1	0.72	0.45	0	0.00	1.0	1.0	1.0	
expenses	0	1	1708.85	7037.01	0	0.00	0.0	0.0	232796.8	

The dataset contains 11 variables and 20,000 client records. The response variable `expenses` is zero-inflated, with many zero values. Predictors include categorical variables (geo_area, vehicle_type),numerical variables (age, est_value, miles_driven) and binary variables (college_grad_ind, late_driving_ind, etc).

3. Convert categorical variables to factors where necessary.

```
autoinsurance<-autoinsurance|>  mutate(
    # Add log_expense
    log_expense = log(expenses),
    # Convert categorical and binary to factors
    geo_area = as.factor(geo_area),
    vehicle_type = case_when(
      vehicle_type %in% c("minivan", "truck") ~ "other",
      TRUE ~ vehicle_type),
    vehicle_type = as.factor(vehicle_type),
    college_grad_ind = as.factor(college_grad_ind),
    speeding_ticket_ind = as.factor(speeding_ticket_ind),
    clean_driving_ind = as.factor(clean_driving_ind),
    hard_braking_ind = as.factor(hard_braking_ind),
    late_driving_ind = as.factor(late_driving_ind))
```

To ensure proper preprocessing and modeling, we convert categorical and binary variables to factors. Additionally, we merge rare levels in vehicle_type, combining "minivan" and "truck" into a single category called "other".

4. Filter out policyholders with zero claim expenses and retain only those with positive expenses.

```
# Filter out zero claims
autoinsurance_filtered <- autoinsurance |> filter(expenses > 0)
# Remove expenses column to avoid leakage
autoinsurance_transformed <- autoinsurance_filtered |> select(-expenses)
```

Since our goal is to predict the positive claim expenses where a claim was made, we filtered out zero-claim cases and log-transformed the response variable to improve model performance.

5. Provide comments on the steps taken.

We loaded and cleaned the dataset, merged rare categories, converted variables to factors, removed zero-claim cases, created a log-transformed response, and dropped the raw `expenses` column to avoid data leakage.

Part (b): Data Splitting

```
# Split data using log_expense as strata
set.seed(47969938)
autoinsurance_split <- initial_split(autoinsurance_transformed, prop = 0.8, strata = log_expe
autoinsurance_train <- training(autoinsurance_split)
autoinsurance_test <- testing(autoinsurance_split)
# Cross-validation folds
autoinsurance_folds <- vfold_cv(autoinsurance_train, v = 10, strata = log_expense)
```

Part (c): Exploratory Data Analysis (EDA)

1. Are there any missing values in the training dataset? If so, how should they be handled?

```
#missing value
colSums(is.na(autoinsurance_train))
```

```
                age            geo_area        vehicle_type           est_value
                  0                   0                   0                   0
      miles_driven    college_grad_ind speeding_ticket_ind    hard_braking_ind
                  0                   0                   0                   0
  late_driving_ind   clean_driving_ind         log_expense
                  0                   0                   0
```

There are no missing values in our training dataset.If there had been any, we could have handled them by either removing rows with missing values, or imputing them using appropriate methods such as the mean or median for numeric variables, or the mode for categorical variables.

2. For categorical variables, do any factor levels have very few observations? Should rare levels be merged?

```
autoinsurance_train |>
  count(geo_area) |>
  arrange(n)
```

```
# A tibble: 3 x 2
  geo_area     n
  <fct>    <int>
1 rural      280
2 urban      481
3 suburban   612
```

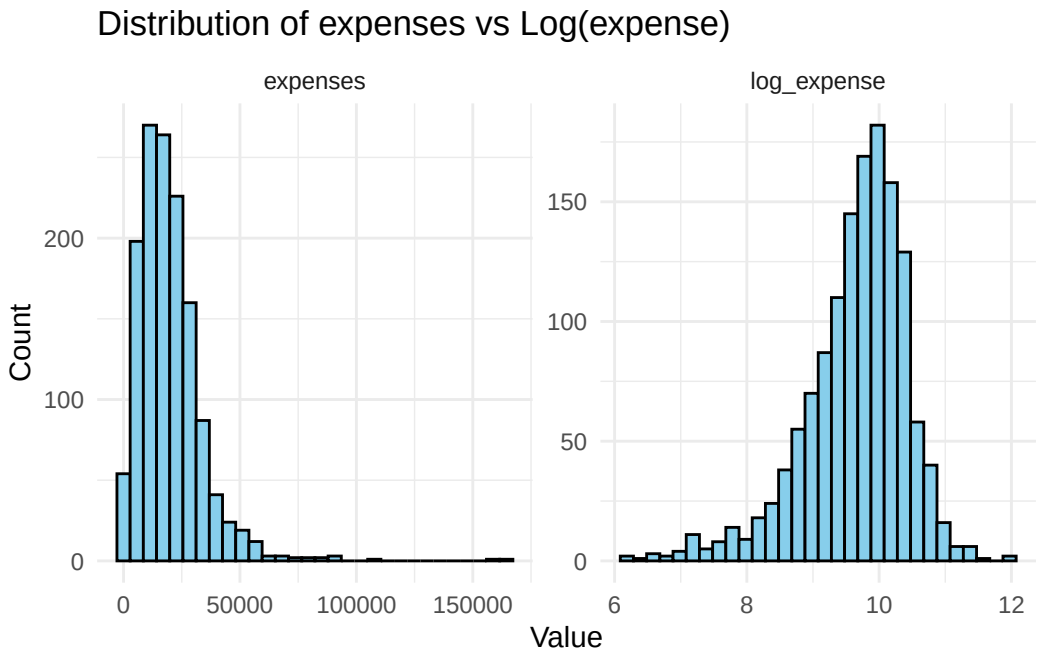
```
autoinsurance_train |>
  count(vehicle_type) |>
  arrange(n)
```

```
# A tibble: 3 x 2
  vehicle_type     n
  <fct>        <int>
1 other          295
2 car            383
3 suv            695
```

We observed that the vehicle_type variable included rare levels such as “minivan” and “truck”. To prevent potential overfitting, these levels were manually grouped into an “other” category during the data preprocessing step using case_when()

3. How is the response variable distributed? Does it require transformation? Justify your decision.

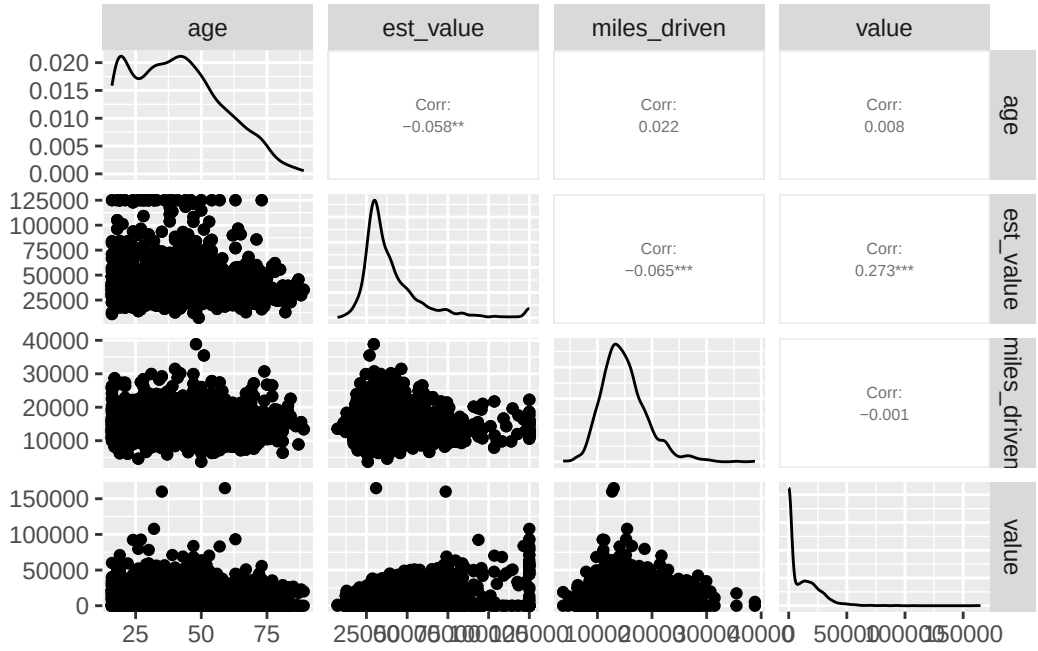
```
autoinsurance_train_eda <- autoinsurance_train |>
  mutate(
    expenses = exp(log_expense)
  ) |>
  pivot_longer(
    cols = c(expenses, log_expense),
    names_to = "type",
    values_to = "value"
  )
ggplot(autoinsurance_train_eda, aes(x = value)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  facet_wrap(~type, scales = "free") +
  labs(title = "Distribution of expenses vs Log(expense)",
       x = "Value", y = "Count") +theme_minimal()
```



The response variable is **highly right-skewed**. To satisfy **regression assumptions** and improve model **stability**, we apply a log transformation. The transformed distribution appears more symmetric, with slight left skewness.To ensure interpretability, the final predictions will be automatically back-transformed to the original scale using exp(), allowing us to evaluate performance based on the actual claim amounts.

4. What are the relationships among the numerical variables? Is there a need to decorrelate them? How do they relate to the response variable? Identify promising predictors.

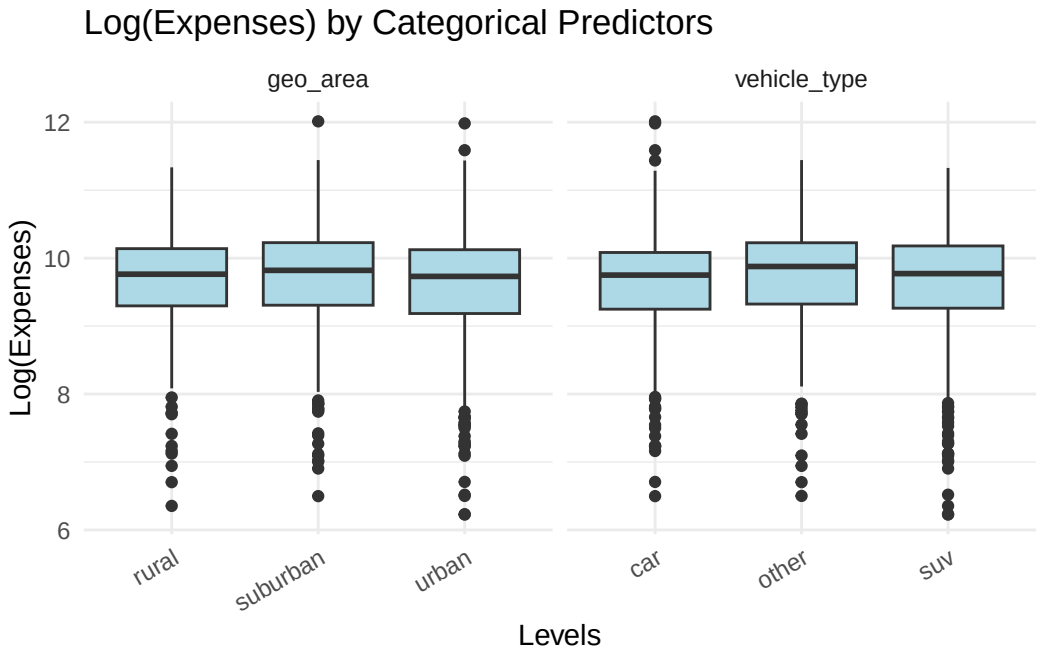
```
library(GGally)
autoinsurance_train_eda |>
  select(where(is.numeric)) |>
  GGally::ggpairs(upper = list(continuous = wrap("cor", size = 2)))
```



All of the numerical predictors have low correlations with each other. The reason we check for decorrelation is to avoid **multicollinearity**, where correlated predictors may provide redundant information. Most variables also show low correlation with the response, possibly because they are coded as binary variables. The only predictor that appears promising is the estimated market value of the vehicle (est_value), which has a moderate correlation with the response (Corr = 0.4).

5. How does the response variable relate to the categorical predictors? Are any of them promising for modelling?

```
autoinsurance_train |>
  select(geo_area, vehicle_type, log_expense) |>
  pivot_longer(cols = -log_expense, names_to = "variable", values_to = "value") |>
  ggplot(aes(x = value, y = log_expense)) +
  geom_boxplot(fill = "lightblue") +
  facet_wrap(~variable, scales = "free_x") +
  labs(x = "Levels", y = "Log(Expenses)",
       title = "Log(Expenses) by Categorical Predictors") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



The boxplots of log(Expenses) across categorical variables show some variation between levels. In particular, vehicle_type appears to be a promising predictor, as the median log(Expenses) differs across types.

Part (d): Model Development and Evaluation

Regularised Regression Model (log_expenses)

```
# Define model
reg_model <- linear_reg(penalty = tune(), mixture = tune()) |>
  set_engine("glmnet") |>
  set_mode("regression")
# Define recipe-----------------------------------------------------------
reg_recipe <- recipe(log_expense ~ ., data = autoinsurance_train) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors())
# Create workflow---------------------------------------------------------
reg_wf <- workflow() |>
  add_model(reg_model) |>
  add_recipe(reg_recipe)
#Grid Search-------------------------------------------------------------
reg_param <- extract_parameter_set_dials(reg_model)
reg_grid <- grid_max_entropy(reg_param, size = 50)
# Tuning (Cross-validation)-----------------------------------------------
set.seed(47969938)
reg_tuned <- tune_grid(
  reg_wf,
  resamples = autoinsurance_folds,
  grid = reg_grid,
  metrics = metric_set(rmse, rsq)
)
#Select Best Parameters--------------------------------------------------
reg_best <- select_best(reg_tuned, metric = "rmse")
final_reg_wf <- finalize_workflow(reg_wf, reg_best)
# Final model fitting----------------------------------------------------
final_reg_fit <- final_reg_wf |> last_fit(autoinsurance_split)
# Evaluate on original scale(on test data)-------------------------------
final_reg_fit |>
  collect_predictions() |>
  mutate(
    .pred = exp(.pred),
    expenses = exp(log_expense)  # Back-transform for comparison
  ) |>
  metrics(truth = expenses, estimate = .pred) |>
  filter(.metric %in% c("rmse", "rsq"))
```

```
# A tibble: 2 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard   16234.
2 rsq     standard       0.126
```

K-Nearest Neighbours Model

```
###### Model Specification
knn_model <- nearest_neighbor(
  neighbors = tune(),
  dist_power = tune(),
  weight_func = tune()
) |>
  set_engine("kknn") |>
  set_mode("regression")
###### recipe  ----------------------------------------------------------------
knn_recipe <- recipe(log_expense ~ ., data = autoinsurance_train) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors())
###### Workflow ---------------------------------------------------------------
knn_wf <- workflow() |>
  add_model(knn_model) |>
  add_recipe(knn_recipe)
###### Grid Search ------------------------------------------------------------
knn_param <- parameters(
  neighbors(),
  dist_power(),
  weight_func()
) |>
  update(
    neighbors = neighbors(range = c(3, 50)),
    dist_power = dist_power(range = c(1, 2)),
    weight_func = weight_func(values = c("rectangular", "inv", "rank", "triweight"))
  )
knn_grid <- grid_regular(knn_param, levels = c(5, 3, 4))
###### Tuning (Cross-validation) ----------------------------------------------
set.seed(47969938)
knn_tuned <- tune_grid(
  knn_wf,
  resamples = autoinsurance_folds,
```

```
  grid = knn_grid,
  metrics = metric_set(rmse, rsq)
)
###### Select Best Parameters ------------------------------------------------
knn_best <- select_best(knn_tuned, metric = "rmse")
###### Finalize Workflow & Fit -----------------------------------------------
final_knn_wf <- finalize_workflow(knn_wf, knn_best)
final_knn_fit <- final_knn_wf |> last_fit(autoinsurance_split)
###### Evaluate on Original Scale --------------------------------------------
final_knn_fit |>
  collect_predictions() |>
  mutate(
    .pred = exp(.pred),
    expenses = exp(log_expense)
  ) |>
  metrics(truth = expenses, estimate = .pred) |>
  filter(.metric %in% c("rmse", "rsq"))
```

```
# A tibble: 2 x 3
  .metric .estimator  .estimate
  <chr>   <chr>           <dbl>
1 rmse    standard   17035.
2 rsq     standard       0.0665
```

Random Forest Model

```
###### Model Specification ---------------------------------------------------
rf_model <- rand_forest(
  mode = "regression",
  mtry = tune(),
  min_n = tune(),
  trees = tune()
) |>
  set_engine("ranger", importance = "impurity")
###### Recipe ----------------------------------------------------------------
rf_recipe <- recipe(log_expense ~ ., data = autoinsurance_train) |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors())
###### Workflow --------------------------------------------------------------
rf_wf <- workflow() |>
```

```r
  add_model(rf_model) |>
  add_recipe(rf_recipe)
###### Grid Search ------------------------------------------------------------
rf_param <- extract_parameter_set_dials(rf_model) |>
  update(
    mtry = mtry(range = c(1L, 8L)),
    min_n = min_n(range = c(2L, 10L)),
    trees = trees(range = c(500L, 1500L))
  )
set.seed(47969938)
rf_grid <- grid_regular(rf_param, levels = 4)
###### Tuning (Cross-validation) ----------------------------------------------
rf_tuned <- tune_grid(
  rf_wf,
  resamples = autoinsurance_folds,
  grid = rf_grid,
  metrics = metric_set(rmse, rsq)
)
###### Select Best Parameter --------------------------------------------------
rf_best <- select_best(rf_tuned, metric = "rmse")
###### Finalize Workflow & Fit ------------------------------------------------
final_rf_wf <- finalize_workflow(rf_wf, rf_best)
final_rf_fit <- final_rf_wf |> last_fit(autoinsurance_split)
###### Evaluate on Original Scale ---------------------------------------------
final_rf_fit |>
  collect_predictions() |>
  mutate(
    .pred = exp(.pred),
    expenses = exp(log_expense)
  ) |>
  metrics(truth = expenses, estimate = .pred) |>
  filter(.metric %in% c("rmse", "rsq"))
```

```
# A tibble: 2 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard   16519.
2 rsq     standard       0.104
```

After tuning, select the best hyperparameter combination for each model, rank models based on RMSE and comment.

```
#best parameters
reg_best
```

```
# A tibble: 1 x 3
  penalty mixture .config
    <dbl>   <dbl> <chr>
1 0.00826   0.595 Preprocessor1_Model32
```

```
knn_best
```

```
# A tibble: 1 x 4
  neighbors weight_func dist_power .config
      <int> <chr>            <dbl> <chr>
1        50 rank                 2 Preprocessor1_Model50
```

```
rf_best
```

```
# A tibble: 1 x 4
   mtry trees min_n .config
  <int> <int> <int> <chr>
1     3  1166    10 Preprocessor1_Model58
```

```
model_ranking <- bind_rows(
  reg_tuned %>%
    show_best(metric = "rmse", n = 1) %>%
    mutate(model = "Regularised Regression (log)"),
  knn_tuned %>%
    show_best(metric = "rmse", n = 1) %>%
    mutate(model = "KNN (log)"),
  rf_tuned %>%
    show_best(metric = "rmse", n = 1) %>%
    mutate(model = "Random Forest (log)")
)
# Show only model and RMSE (mean)
model_ranking %>%
  select(model, mean) %>%
  rename(rmse = mean) %>%
  arrange(rmse)
```

```
# A tibble: 3 x 2
  model                          rmse
  <chr>                         <dbl>
1 Regularised Regression (log) 0.711
2 Random Forest (log)          0.723
3 KNN (log)                    0.739
```

Based on the RMSE values, the Regularised Regression (log) model outperforms the other models, indicating it provides the most accurate overall predictions on the original expense scale.

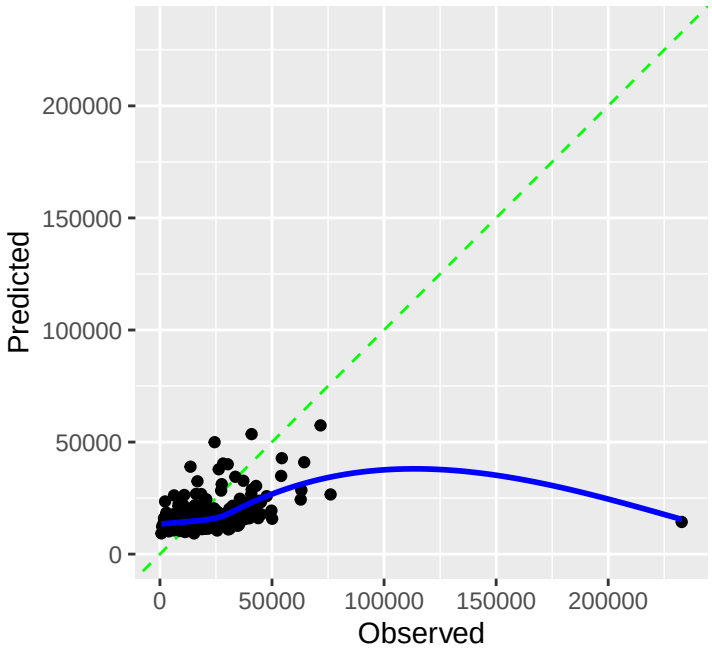
Part (e):

Report the test set RMSE for the best-performing model

```r
#calibration plot
library(probably)
set.seed(47969938)
autoinsurance_split <- initial_split(autoinsurance_filtered, prop = 0.8, strata = log_expense
autoinsurance_train <- training(autoinsurance_split)
autoinsurance_test <- testing(autoinsurance_split)
final_reg_model <- finalize_workflow(reg_wf, reg_best)
final_reg_fit <- fit(final_reg_model, data = autoinsurance_train)
reg_augmented <- augment(final_reg_fit, new_data = autoinsurance_test)
reg_augmented |>
  mutate(
    predicted_expenses = exp(.pred),
    actual_expenses = expenses
  ) |>
  metrics(truth = actual_expenses, estimate = predicted_expenses) |>
  filter(.metric %in% c("rmse", "rsq"))
```

```
# A tibble: 2 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard   16234.
2 rsq     standard       0.126
```

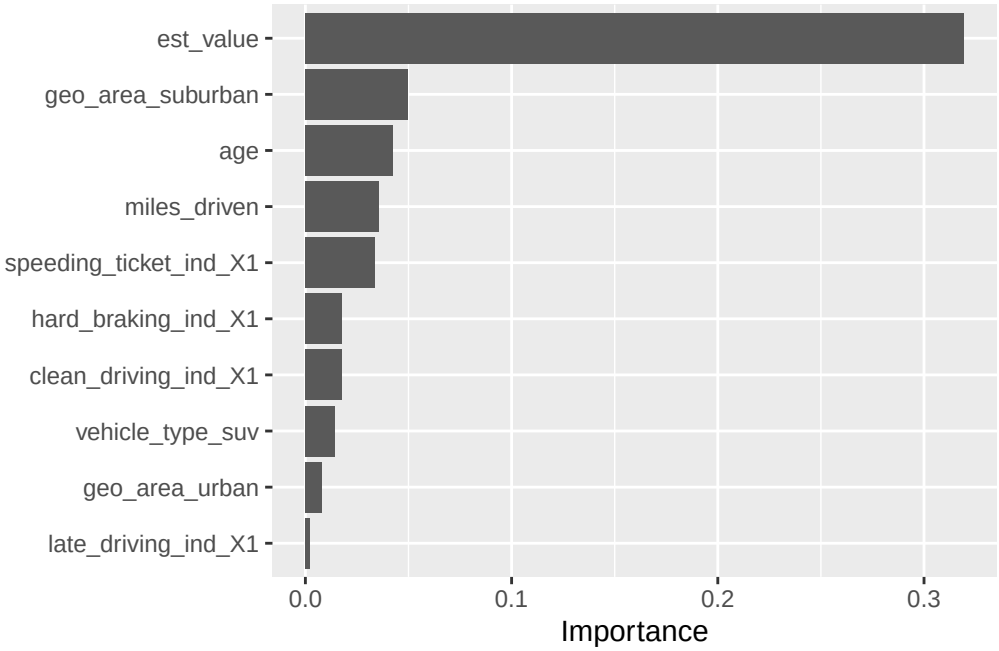
```r
reg_augmented |>
  mutate(
    predicted_expenses = exp(.pred)
  ) |>
  probably::cal_plot_regression(
    truth = expenses,
    estimate = predicted_expenses
  )
```



The best-performing regularised regression model achieved an RMSE of approximately 16,234, which means that on average, the model's predictions deviate from the actual claim expenses by around 16,234 units on the original scale (after back-transforming from the log scale). While this indicates a reasonable predictive ability given the wide range of expenses, there is still room for improvement, especially in capturing higher expense cases as seen in the calibration plot.

Identify the most important predictors based on the final model

```r
library(vip)
final_reg_fit %>%
  vip(num_features = 10)
```



Based on the variable importance plot from the final regularised regression model, the most important predictor was the vehicle's estimated market value, as higher-value cars tend to have higher claim expenses. Living in suburban areas and driving longer distances also appeared influential, likely reflecting different driving environments and higher exposure to risks. The policyholder's age and having a history of speeding tickets were also key factors, pointing to varying risk profiles and driving behaviours. Other moderately important variables included indicators of driving behaviour like hard braking and clean driving, as well as vehicle type and living in urban areas.

Compare the findings with your Exploratory Data Analysis (EDA) from Part (c)

The patterns we saw in the EDA generally align with the final model results. The strong impact of est_value in the model confirms what we observed earlier, where higher vehicle values were linked to higher claim expenses. Similarly, age and miles_driven showed clear patterns in the EDA, with older vehicles and higher mileage associated with increased costs. The significance of suburban areas also fits with our earlier findings, as suburban drivers tended to have slightly higher claims. Risk indicators like speeding tickets and hard braking, which suggested higher risks during the EDA, were also confirmed as important predictors in the model. Overall, the final model reinforced our initial observations and gave stronger evidence for the importance of these variables.

Part (f):

There were two major challenges I encountered while constructing the models.

The first challenge was deciding whether or not to apply a log transformation to the response variable (expenses). While I already knew that regularised regression benefits from a log transformation due to its linear assumptions, I wasn't sure whether KNN and Random Forest should also be transformed. This led me to consider whether all models needed to be trained on the same transformed scale, or whether it would be acceptable to only apply the transformation to some of them. I also had to think carefully about how to compare model performance fairly on the original expense scale, especially if some models were trained using transformed values.

The second challenge involved deciding how to perform grid search for each model. Although I read documentation from the tidymodels website and other online sources, I still struggled with choosing appropriate tuning methods and parameter ranges for different types of models. This was especially difficult for tree-based models, where the interaction between hyperparameters is complex.

To explore the transformation issue further, I created both log-transformed and non-transformed versions of KNN and Random Forest models. Even though tuning these models took extra time and made me feel a bit overwhelmed, the experience taught me that log transformation can improve prediction accuracy for some models (like RRM), but not necessarily for others. In my early version, I forgot to convert predictions back from the log scale before calculating RMSE, which led to inaccurate model comparisons. From this, I learned the importance of always evaluating models on the same scale.

The other thing is that I realized it's completely fine to apply transformations to the target variable—like log—as long as we transform the predictions back and the evaluation metrics such as RMSE and R^2 are acceptable on the original scale. This helped me understand that the key is not whether we transform the target or not, but whether the final predictions are accurate and meaningful in their real-world context.

Overall, this experience deepened my understanding of how modelling decisions—such as pre-processing choices and tuning strategies—interact with both the data and the algorithms. It reinforced the importance of thoughtful design and critical evaluation in the predictive modelling process.