Class Activity 23

Your name here

May 17 2024

Group Activity 1

Load the `mlbench` package to get `PimaIndiansDiabetes2` dataset.

```r
set.seed(507581761)
# Load the data - diabetes
data(PimaIndiansDiabetes2)
db <- PimaIndiansDiabetes2
db <- db %>% drop_na()
db_raw <- db %>% select(glucose, insulin, diabetes)

db_split <- initial_split(db_raw, prop = 0.80)
# Create training data
db_train <- db_split %>% training()
# Create testing data
db_test <- db_split %>% testing()
```

a. *Creating the Recipe:* Construct a recipe for the model by normalizing `glucose` and `insulin` predictors to predict `diabetes` status on the training set, ensuring data scales are comparable.

```r
db_recipe <- recipe(diabetes ~ glucose + insulin, data = db_train) %>%
  step_scale(all_predictors()) %>%
  step_center(all_predictors())
```

b. *Model Specification:* Define the KNN model using a flexible `tune()` placeholder for the number of neighbors, specifying a `classification` task.

```r
knn_spec <- nearest_neighbor(weight_func = "rectangular",
                             engine = "kknn",
                             mode = "classification",
                             neighbors = tune())
```

c. *Creating Folds:* Divide the training data into 10 stratified folds based on the diabetes outcome to prepare for cross-validation, ensuring representation.

```r
db_vfold <- vfold_cv(db_train, v = 10, strata = diabetes, repeats = 10)
```

d. *Cross-Validation Grid:* Generate a sequence of K values to test with 10-fold cross-validation, evaluating model performance across a range of neighbors.

```r
k_vals <- tibble(neighbors = seq(from = 1, to = 40, by = 1))
```

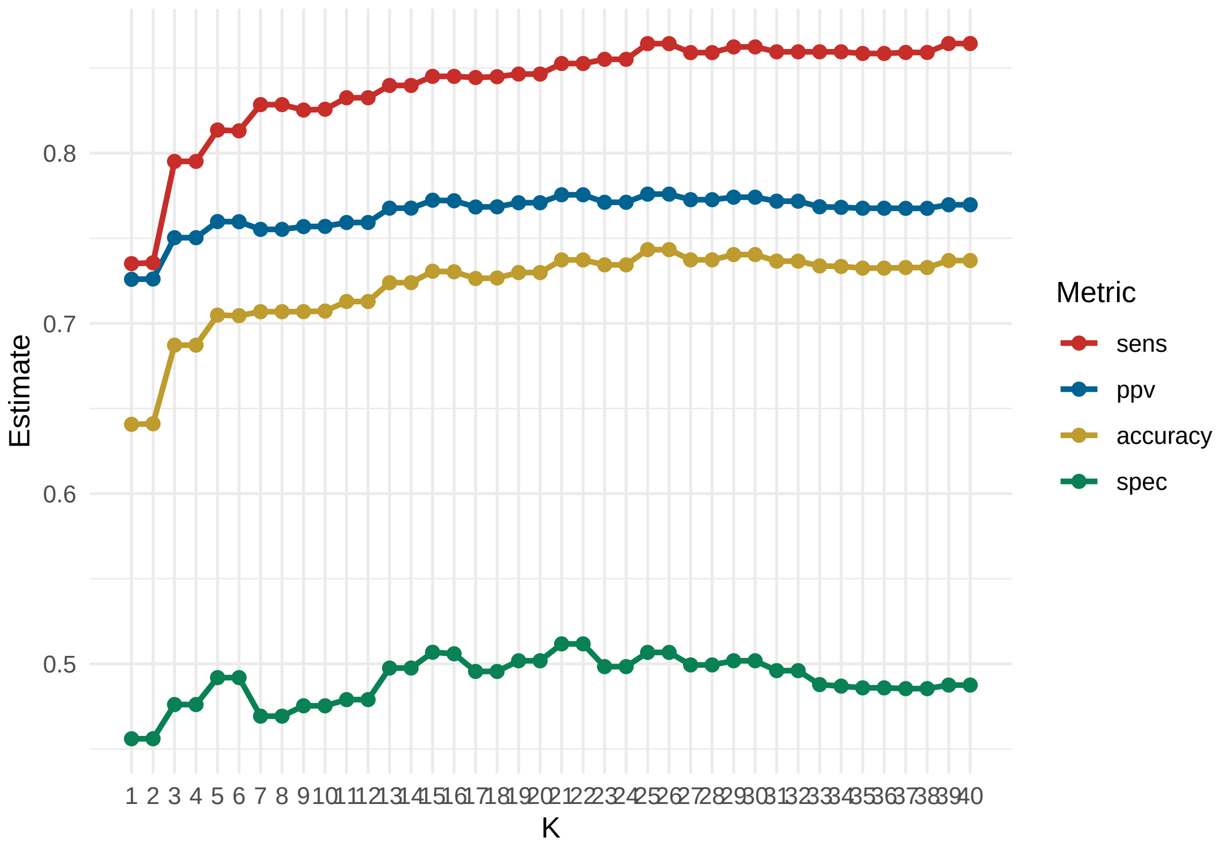
```r
knn_fit <- workflow() %>%
  add_recipe(db_recipe) %>%
  add_model(knn_spec) %>%
  tune_grid(
    resamples = db_vfold,
    grid = k_vals,
    metrics = metric_set(yardstick::ppv, yardstick::accuracy, sens, spec),
    control = control_resamples(save_pred = TRUE))
```

```r
cv_metrics <- collect_metrics(knn_fit)
cv_metrics
# A tibble: 160 x 7
   neighbors .metric  .estimator  mean     n std_err .config
       <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
 1         1 accuracy binary     0.641   100 0.00815 Preprocessor1_Model01
 2         1 ppv      binary     0.726   100 0.00595 Preprocessor1_Model01
 3         1 sens     binary     0.735   100 0.00985 Preprocessor1_Model01
 4         1 spec     binary     0.456   100 0.0138  Preprocessor1_Model01
 5         2 accuracy binary     0.641   100 0.00810 Preprocessor1_Model02
 6         2 ppv      binary     0.726   100 0.00594 Preprocessor1_Model02
 7         2 sens     binary     0.736   100 0.00979 Preprocessor1_Model02
 8         2 spec     binary     0.456   100 0.0138  Preprocessor1_Model02
 9         3 accuracy binary     0.687   100 0.00647 Preprocessor1_Model03
10         3 ppv      binary     0.750   100 0.00509 Preprocessor1_Model03
# i 150 more rows
```

e. *Visualization:* Plot the cross-validation results to determine the optimal K value, comparing different performance metrics visually.

The optimal value is around 25

```r
final.results <- cv_metrics %>%  mutate(.metric = as.factor(.metric)) %>%
  select(neighbors, .metric, mean)

final.results %>%
  ggplot(aes(x = neighbors, y = mean, color = forcats::fct_reorder2(.metric, neighbors, mean))) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  theme_minimal() +
  scale_color_wsj() +
  scale_x_continuous(breaks = k_vals[[1]]) +
  theme(panel.grid.minor.x = element_blank())+
  labs(color='Metric', y = "Estimate", x = "K")
```



Group Activity 2

a. Data Preparation and Train-Test Split

Load the `mlbench` package and `tidymodels` framework, select relevant features for predicting `glucose`, and split the data into training and test sets. For this activity, use `mass` and `insulin` as your features.

```r
library(mlbench)
library(tidymodels)
library(dplyr)

data(PimaIndiansDiabetes2)
db <- PimaIndiansDiabetes2 %>%
  drop_na() %>%
  select(glucose, mass, insulin)

# Splitting the data
set.seed(2056)
db_split <- initial_split(db, prop = 0.75)
db_train <- training(db_split)
db_test <- testing(db_split)
```

b. Model Specification

Define a linear regression model for predicting `glucose` as a function of `mass` and `insulin`.

```r
lm_spec <- linear_reg() %>%
  set_engine(engine = "lm") %>%
  set_mode("regression")
```

c. Fit the Model

Fit the linear model to the training data, predicting `glucose` based on `mass` and `insulin`.

```
lm_mod <- lm_spec %>%
  fit(glucose ~ ., data = db_train)
```

d. Predict on Test Data and Evaluate the Model

Use the fitted model to predict `glucose` levels on the test set and evaluate the model's accuracy with RMSE and R-squared metrics.

```
# Predicting glucose levels
results <- db_test %>%
  bind_cols(predictions = predict(lm_mod, new_data = db_test, type = "raw")) %>%
  select(glucose, predictions)

# Displaying first 6 predictions
results %>%
  slice_head(n = 6) %>%
  knitr::kable()
```

	glucose	predictions
4	89	110.36355
17	118	141.12540
44	171	142.31251
51	103	103.14016
53	88	97.80386
60	105	125.73907

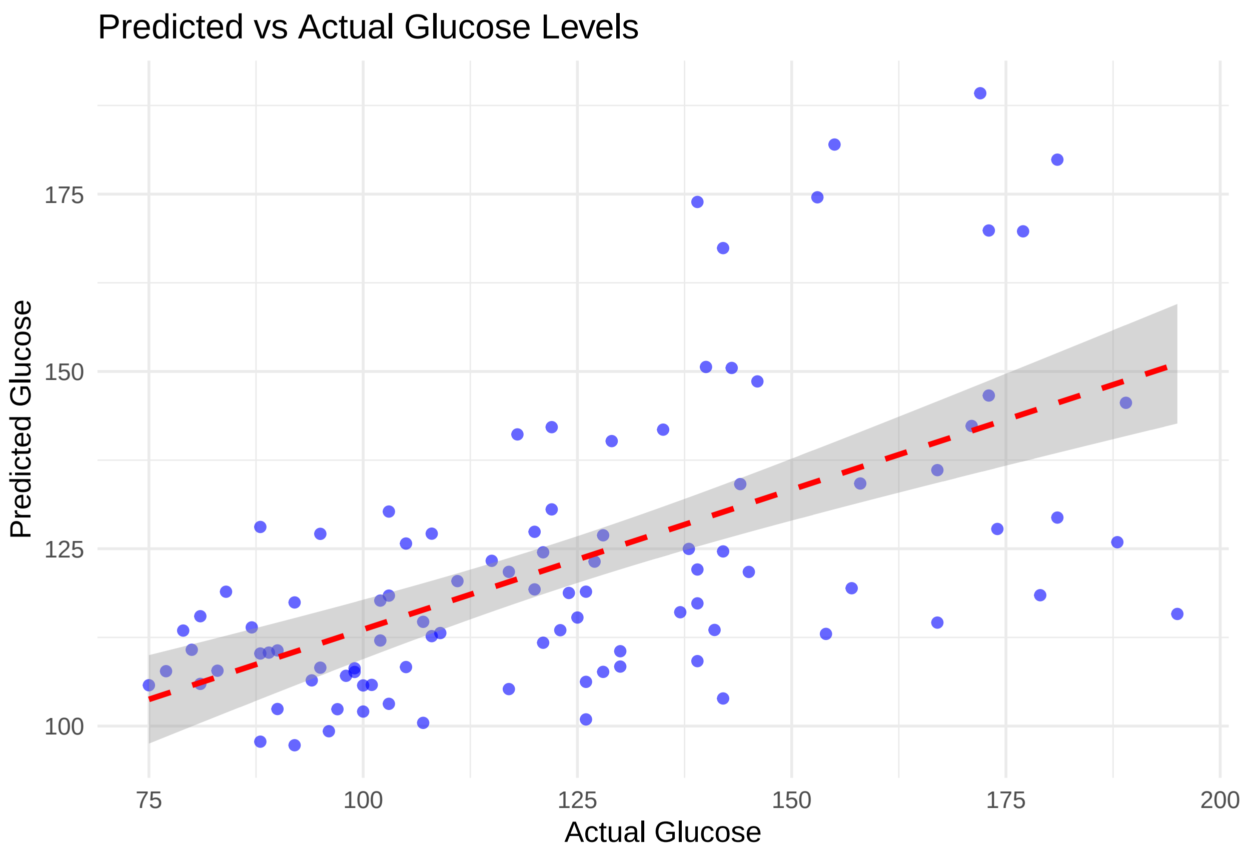
```
# Evaluating the model
eval_metrics <- metric_set(rmse, rsq)

eval_metrics(data = results,
             truth = glucose,
             estimate = predictions) %>%
  #select(-2) %>%
  knitr::kable()
```

.metric	.estimator	.estimate
rmse	standard	24.50393
rsq	standard	0.35043

(Bonus) Create a scatter plot to visualize the actual vs. predicted glucose levels, including a regression line for reference.

```
results %>%
  ggplot(aes(x = glucose, y = predictions)) +
```

```r
  geom_point(color = "blue", alpha = 0.6) +
  geom_smooth(method = "lm", color = "red", linetype = "dashed") +
  labs(title = "Predicted vs Actual Glucose Levels",
       x = "Actual Glucose",
       y = "Predicted Glucose") +
  theme_minimal()
```


Predicted vs Actual Glucose Levels
Predicted Glucose
Actual Glucose