

t-SNE with SVM Regression and Stability Check

OBJECTIVE

To evaluate regression performance using SVM on t-SNE-reduced features of the mtcars dataset and assess t-SNE embedding stability using clustering quality.

LIBRARIES REQUIRED

- **caret** – Model training and cross-validation
- **e1071** – SVM support
- **Rtsne** – t-SNE dimensionality reduction
- **cluster** – Silhouette analysis
- **ggplot2** – Data visualization

STEPS

1. Load mtcars and inspect structure
2. Apply t-SNE for dimensionality reduction
3. Split data into training and testing sets
4. Train SVM with hyperparameter tuning via cross-validation
5. Predict on test data and compute RMSE
6. Visualize t-SNE results (scatter & density plots)
7. Run t-SNE multiple times, apply k-means, and compute silhouette scores for stability

CODE SNIPPET

```
install.packages("caret", repos='https://cloud.r-project.org')  
  
## package 'caret' successfully unpacked and MD5 sums checked  
##
```

```
## The downloaded binary packages are in
## C:\Users\Arun Santhosh R A\AppData\Local\Temp\Rtmpgbltd\downloaded_packages
```

```
install.packages("e1071", repos='https://cloud.r-project.org')
```

```
## package 'e1071' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Arun Santhosh R A\AppData\Local\Temp\Rtmpgbltd\downloaded_packages
```

```
install.packages("Rtsne", repos='https://cloud.r-project.org')
```

```
## package 'Rtsne' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Arun Santhosh R A\AppData\Local\Temp\Rtmpgbltd\downloaded_packages
```

```
install.packages("cluster", repos='https://cloud.r-project.org')
```

```
## package 'cluster' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Arun Santhosh R A\AppData\Local\Temp\Rtmpgbltd\downloaded_packages
```

```
install.packages("ggplot2", repos='https://cloud.r-project.org')
```

```
## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Arun Santhosh R A\AppData\Local\Temp\Rtmpgbltd\downloaded_packages
```

```
library(caret)
library(e1071)
library(Rtsne)
library(cluster)
library(ggplot2)
```

```
# Load the mtcars dataset
data(mtcars)
```

```
# Inspect structure
str(mtcars)
```

```
## 'data.frame': 32 obs. of 11 variables:
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...
## $ disp: num 160 160 108 258 360 ...
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...
```

```
## $ qsec: num 16.5 17 18.6 19.4 17 ...
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...
```

```
# Print number of attributes and instances
cat("Number of attributes:", ncol(mtcars), "\n")
```

```
## Number of attributes: 11
```

```
cat("Number of instances:", nrow(mtcars), "\n")
```

```
## Number of instances: 32
```

```
# Perform t-SNE
set.seed(123)
tsne_result <- Rtsne(mtcars, dims = 2, perplexity = 5, verbose = TRUE)
```

```
## Performing PCA
## Read the 32 x 11 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 5.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.00 seconds (sparsity = 0.539062)!
## Learning embedding...
## Iteration 50: error is 53.811003 (50 iterations in 0.00 seconds)
## Iteration 100: error is 52.492368 (50 iterations in 0.00 seconds)
## Iteration 150: error is 55.397535 (50 iterations in 0.00 seconds)
## Iteration 200: error is 58.238593 (50 iterations in 0.00 seconds)
## Iteration 250: error is 54.714276 (50 iterations in 0.00 seconds)
## Iteration 300: error is 1.530693 (50 iterations in 0.00 seconds)
## Iteration 350: error is 0.996892 (50 iterations in 0.00 seconds)
## Iteration 400: error is 0.380054 (50 iterations in 0.00 seconds)
## Iteration 450: error is 0.428135 (50 iterations in 0.00 seconds)
## Iteration 500: error is 0.342240 (50 iterations in 0.00 seconds)
## Iteration 550: error is 0.156026 (50 iterations in 0.00 seconds)
## Iteration 600: error is 0.144303 (50 iterations in 0.00 seconds)
## Iteration 650: error is 0.135936 (50 iterations in 0.00 seconds)
## Iteration 700: error is 0.138272 (50 iterations in 0.00 seconds)
## Iteration 750: error is 0.132766 (50 iterations in 0.00 seconds)
## Iteration 800: error is 0.124820 (50 iterations in 0.00 seconds)
## Iteration 850: error is 0.125517 (50 iterations in 0.00 seconds)
## Iteration 900: error is 0.126720 (50 iterations in 0.00 seconds)
## Iteration 950: error is 0.117548 (50 iterations in 0.00 seconds)
## Iteration 1000: error is 0.121294 (50 iterations in 0.00 seconds)
## Fitting performed in 0.02 seconds.
```

```
# Create a data frame with t-SNE embeddings and mpg
tsne_df <- data.frame(Y1 = tsne_result$Y[,1], Y2 = tsne_result$Y[,2], mpg = mtcars$mpg)
```

```

# Split t-SNE data into training and testing sets
set.seed(123)
train_indices <- createDataPartition(tsne_df$mpg, p = 0.7, list = FALSE)
train_data <- tsne_df[train_indices, ]
test_data <- tsne_df[-train_indices, ]

# Define hyperparameter grid for SVM radial kernel
svm_grid <- expand.grid(sigma = c(0.1, 1, 10),
                        C = c(0.1, 1, 10))

# Train SVM with cross-validation
svm_model <- train(
  mpg ~ .,
  data = train_data,
  method = "svmRadial",
  trControl = trainControl(method = "cv", number = 5),
  tuneGrid = svm_grid
)
svm_model

```

```
## Support Vector Machines with Radial Basis Function Kernel
```

```
##
```

```
## 24 samples
```

```
## 2 predictor
```

```
##
```

```
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
```

```
## Summary of sample sizes: 18, 19, 20, 19, 20
```

```
## Resampling results across tuning parameters:
```

```
##
```

##	sigma	C	RMSE	Rsquared	MAE
##	0.1	0.1	4.748368	0.7635709	3.736303
##	0.1	1.0	3.261250	0.7875730	2.455400
##	0.1	10.0	2.757809	0.8072369	2.154735
##	1.0	0.1	4.780475	0.6651865	3.690169
##	1.0	1.0	2.797225	0.8212324	2.130780
##	1.0	10.0	3.179952	0.8396731	2.694109
##	10.0	0.1	5.406521	0.6407525	4.282925
##	10.0	1.0	3.271162	0.6756121	2.888282
##	10.0	10.0	3.113474	0.6739416	2.751812

```
##
```

```
## RMSE was used to select the optimal model using the smallest value.
```

```
## The final values used for the model were sigma = 0.1 and C = 10.
```

```
# Predict on test set
```

```
predicted_values <- predict(svm_model, newdata = test_data)
```

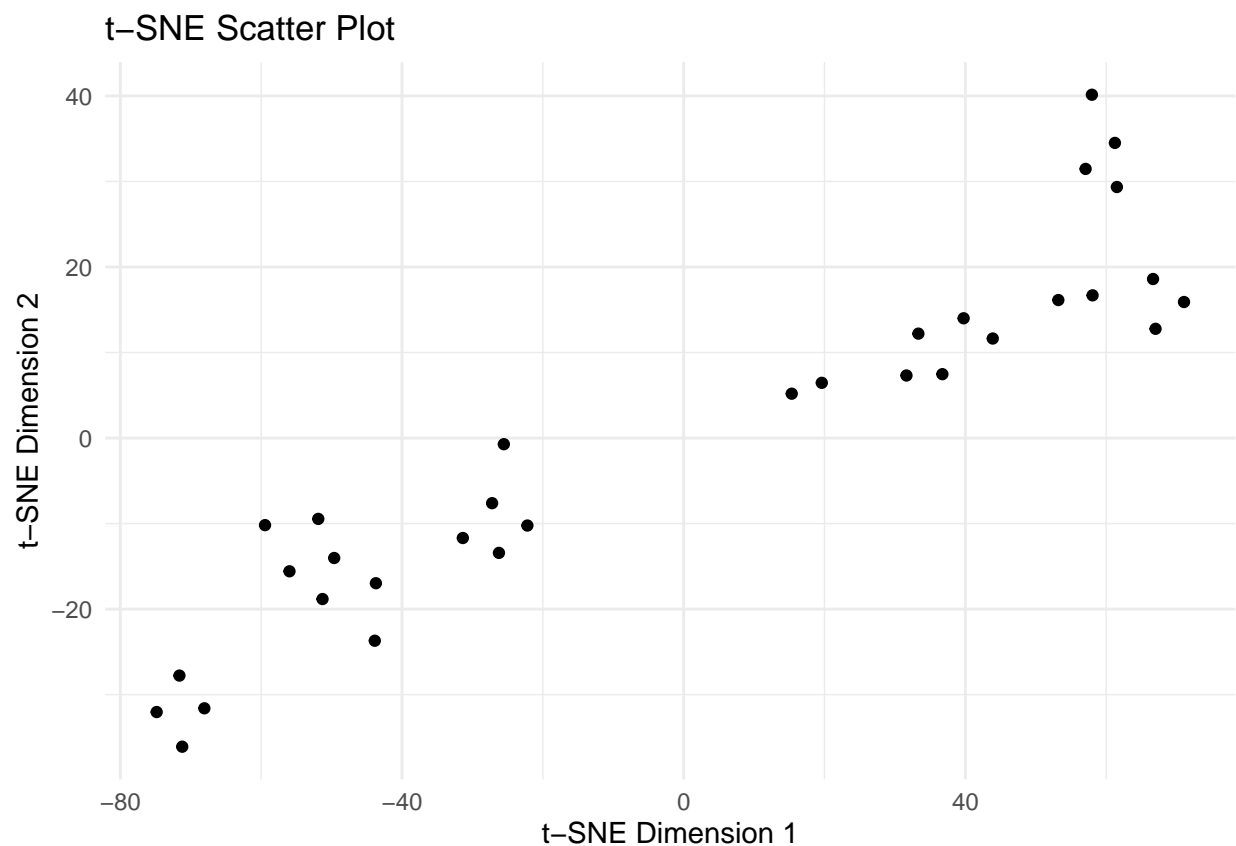
```
# Calculate RMSE
```

```
rmse <- sqrt(mean((predicted_values - test_data$mpg)^2))
```

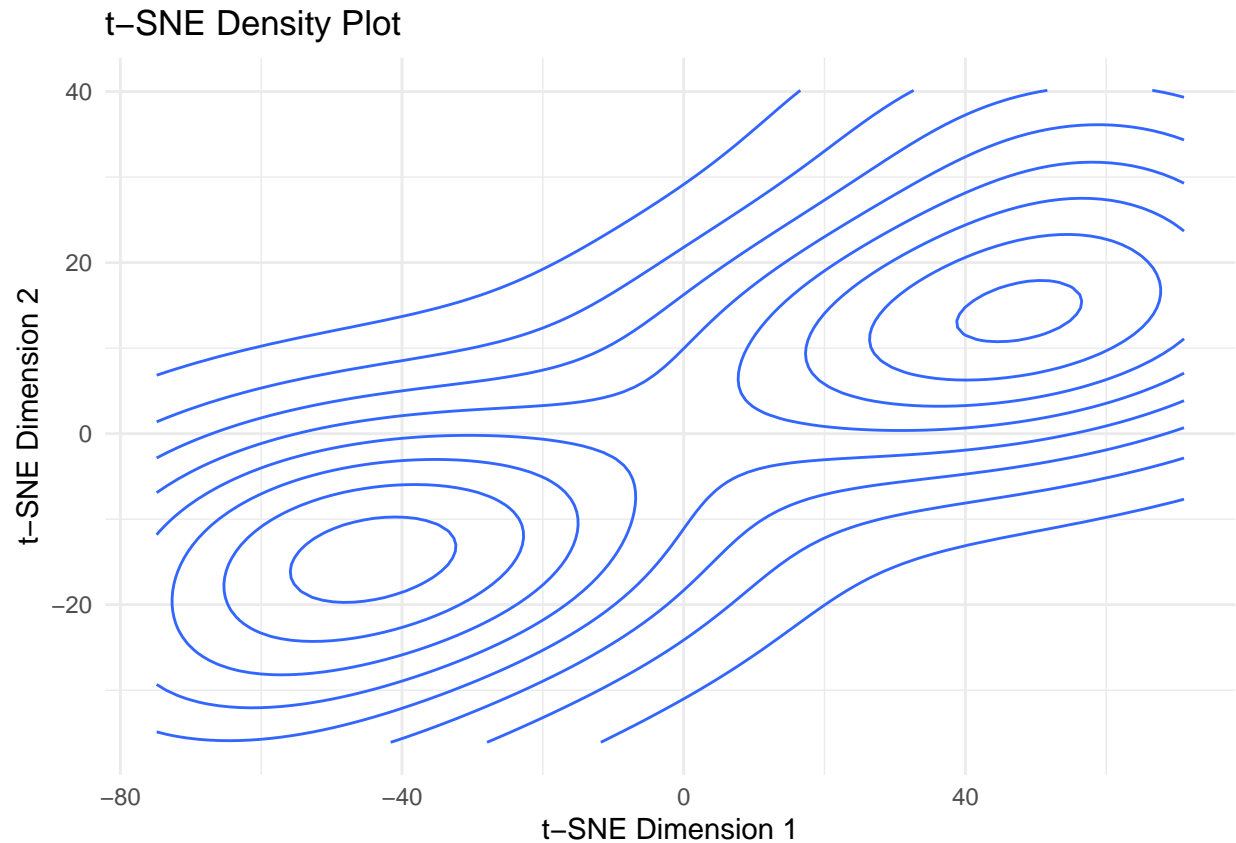
```
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 3.365478
```

```
# Scatter plot of t-SNE results
scatter_plot <- ggplot(tsne_df, aes(x = Y1, y = Y2)) +
  geom_point() +
  labs(title = "t-SNE Scatter Plot",
       x = "t-SNE Dimension 1",
       y = "t-SNE Dimension 2") +
  theme_minimal()
print(scatter_plot)
```



```
# Density plot of t-SNE results
density_plot <- ggplot(tsne_df, aes(x = Y1, y = Y2)) +
  geom_density_2d() +
  labs(title = "t-SNE Density Plot",
       x = "t-SNE Dimension 1",
       y = "t-SNE Dimension 2") +
  theme_minimal()
print(density_plot)
```



Stability analysis of t-SNE embeddings over multiple runs

```
set.seed(123)
n_runs <- 5
sil_scores <- numeric(n_runs)

for (i in 1:n_runs) {
  tsne_tmp <- Rtsne(mtcars, dims = 2, perplexity = 5, verbose = FALSE)
  df_tmp <- data.frame(tsne_tmp$Y)
  kmeans_tmp <- kmeans(df_tmp, centers = 3)
  sil_tmp <- silhouette(kmeans_tmp$cluster, dist(df_tmp))
  sil_scores[i] <- mean(sil_tmp[, 3])
}

cat("Silhouette scores over runs:", round(sil_scores, 3), "\n")
```

```
## Silhouette scores over runs: 0.593 0.579 0.643 0.556 0.605
```

```
cat("Mean Silhouette:", round(mean(sil_scores), 3), " SD:", round(sd(sil_scores), 3), "\n")
```

```
## Mean Silhouette: 0.595 SD: 0.033
```

CONCLUSION

t-SNE effectively reduced data to 2D for regression and visualization. SVM regression achieved reasonable RMSE. Silhouette score analysis showed moderate stability of t-SNE embeddings across runs, highlighting the importance of evaluating t-SNE randomness in downstream tasks.