Final Project

Farook Ahmed Ali Shaik

2024-04-15

Introduction [15 points]

What research question(s) would you like to answer?

Descriptive Statistics:

Q1: How does the computational workload on the Cori supercomputer exhibit distinct patterns or variations throughout the week?

Predictive Modeling:

Q2: What is the optimal predictive model for forecasting power usage of future jobs and applications on the NERSC Cori supercomputer?

Correlation Analysis:

Q3: What are the correlations between power consumption, memory usage, and other variables on the Cori cluster, and how can these correlations inform strategies to optimize energy efficiency?

Strategy Development:

Q4: How does the type of application influence resource consumption in terms of power, memory usage, and computational load in large scale computing environments like Cori?

Q5: Are there any particular areas in any need of strategies for improvement in optimal resource utilization by identifying high utilization or intensive periods on NERSC Cori?

Why a data-driven, computational approach may be useful to answer the questions?

The computational approach is efficient to answer the research questions related to variables that are affecting power consumption in a NERSC Cori supercomputer. Because:

- Analysing the data with the traditional approach is difficult if the data is too large/vast, complex and consist complex relationships, so computational appraoch is useful and gives a systematic approach.
- Machine learning algorithms are efficient in predicting the future outcomes and indentifying in complex relationships between the varibles
- Computational approach also includes of visualisation graphs that are really useful in finding out hidden patterns, outliers, data skewness and etc.

Because of these advantages, computational approach is useful to predict the future power consumption and analyze the patterns and performances of different factors that are effecting in power consumption in a NERSCE Cori supercomputer.

Describe the dataset that you choose.

The dataset focuses on NERSC Cori, a powerful supercomputer at the Department of Energy's National Energy Research Scientific Computing Center. It details power consumption in 2020 and usage patterns across various applications. However, my project focuses on One Week Jobs Data which has 4,401 observations and 14 columns and breif description of these columns is as follows:

Categorical Variables:

exename:- Name of the executable applications being run for the job.

appname:- Identifying name of the application.

Datetime Variables:

start:- Start time of the job.

end:- Completion time of the job.

Numerical Variables:

jobid, numnodes, numcpus, numtasks, executiontime, power, memory, taskspercpu, tasksperpnode, user

Computational Methods [30 points]

For the choosen dataset, what are the necessary data wrangling steps to make the data ready for subsequent analyses?

To make the data tidy and ready for subsequent analyses the steps are as follows:

- Load all ncecessary libraries and packages
- Import the data using read_csv()
- Identify and handle missing values, outliers and etc.
- Check all columns data types and convert them to required datatype.
- Convert the units/metrics of columns if necessary for better analyses.

What exploratory analyses and modeling techniques can be used to answer the research questions?

Descriptive Statistics:

- Plot bar chart to visualise the each day execution time trend
- Use heat map to visualise the major execution time hours
- check for variations using both visualisations

Predictive Modeling:

- split the entire data into train,validation,and test data sets,Because an observation can be used multiple times for exploratory analysis, but only once for confirmatory analysis.
- Develop reression models (linear model, polynomial regression, random forest) and choose optimal model for prediction
- Choosing a suitable machine learning model for power consumption prediction according to the analyses and extracted insights.

Correlation Analysis:

- Calculate the correlation between power and other variables
- Visualise the relation using heatmaps from correlation matrix

Strategy Development:

- Calculate the resource consumption (power,memory,cpu,etc) per each application and see the influence of application and treat the applications accordingly in optimal resource utilisation
- Identify the cpu and memory utilisation over time and visualise using line plots and bar plots and from the obtained insights generate strategies for optimal resource utilisation and find out the high utilisation periods accordingly

What metrics will be used to evaluate the quality of the data analysis?

For **Descriptive Statistics** looking the presence of clear trends or patterns in visualisations can indicate that the data is of good quality and the analysis is accurate.

For **Predicitive Modeling**:

Cross Validation, Mean Cross-Validated (RMSE), Mean Squared Error(MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) are used to evaluate the quality of data analyses and models performance.

We need to choose the model that has the lowest error metrics(MSE,RMSE,MAE). Because the lower these values the better performs the model in detection of underlying distribution in data (either in prediction or minimal error in predicted values and actual values).

However, it's also important to consider the complexity of the model and the risk of overfitting. A more complex model might have slightly lower error metrics, but if it's too complex, it might not generalize well to new data. This is where cross-validation can be very helpful, as it gives a more robust estimate of model performance.

For **Correlation Analysis** visual inspection of heat map of correlation matrix evaluated the quality of data analyses

For **Strategy Development** visual inspection of line plots and bar plots and having thorough understanding previously obtained insights can give most reliable development strategies

This way various metrics comes into play according to the nature of the research question

Data Analysis and Results [40 points]

Perform data analysis, document the analysis procedure, and evaluate the outcomes.

Present the data analysis results.

Interpret the results in a way to address the research questions.

```
library(tidyverse)
library(lubridate)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 4.3.3
```

```r
library(cowplot)
```

```
## Warning: package 'cowplot' was built under R version 4.3.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3
```

```r
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.3.3
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3
```

```r
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 4.3.3
```

```r
library(broom)
```

```r
setwd("D:/C.S KSU/FINAL PROJECT")
data <- read_csv("Cori2020_SelectApps_OneWeek_JobsData.csv")

#Converting the variables 'exename' and 'appname' into factor variables and removing NA values if prese

data_tidy <- data |>
  mutate(exename = as.factor(exename)) |>
  mutate(appname = as.factor(appname)) |>
  na.omit()

#Displaying the first 7 rows of the data
head(data_tidy,5)
```

```
## # A tibble: 5 x 14
##   jobid exename  numnodes numcpus numtasks executiontime start
##   <dbl> <fct>       <dbl>   <dbl>    <dbl>         <dbl> <dttm>
## 1     1 vasp_std        1     256       64          1234 2020-11-30 02:10:50
## 2     2 vasp_std        2     512      128           694 2020-11-30 13:56:05
## 3     3 vasp_std        2     512      128          1046 2020-11-30 13:38:42
## 4     4 vasp_std        1     256       64          3306 2020-11-30 16:59:15
## 5     5 vasp_std        1     256       64          3364 2020-12-01 04:08:45
## # i 7 more variables: end <dttm>, power <dbl>, memory <dbl>, appname <fct>,
## #   taskspercpu <dbl>, taskspernode <dbl>, user <chr>
```

```r
#Displaying the last 7 rows of the data
tail(data_tidy,5)
```

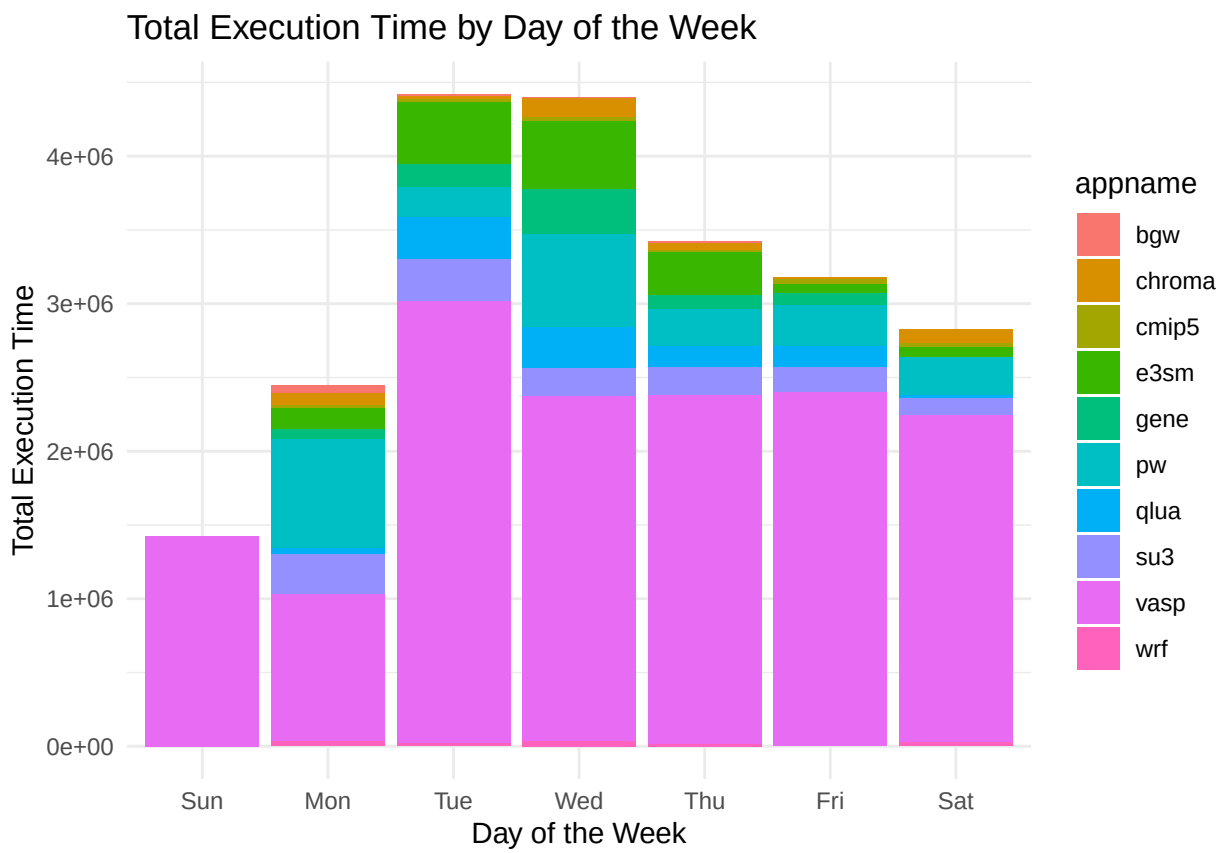
```
## # A tibble: 5 x 14
##   jobid exename      numnodes numcpus numtasks executiontime start
##   <dbl> <fct>           <dbl>   <dbl>    <dbl>         <dbl> <dttm>
## 1  4397 global_fcst~       80   21760       80          1838 2020-12-04 22:12:33
## 2  4398 global_fcst~       80   21760       80          1861 2020-12-05 05:14:18
## 3  4399 global_fcst~       80   21760       80          2070 2020-12-05 05:18:32
## 4  4400 global_fcst~       80   21760       80          2061 2020-12-05 07:38:20
## 5  4401 global_fcst~       80   21760       80          2024 2020-12-05 11:54:21
## # i 7 more variables: end <dttm>, power <dbl>, memory <dbl>, appname <fct>,
## #   taskspercpu <dbl>, taskspernode <dbl>, user <chr>
```

```r
# Splitting the dat into trian, validate, and test data sets
set.seed(24)
pc <- initial_validation_split(data_tidy, prop = c(0.6, 0.2))
train_data <- training(pc)
validation_data <- validation(pc)
test_data <- testing(pc)
pc
```

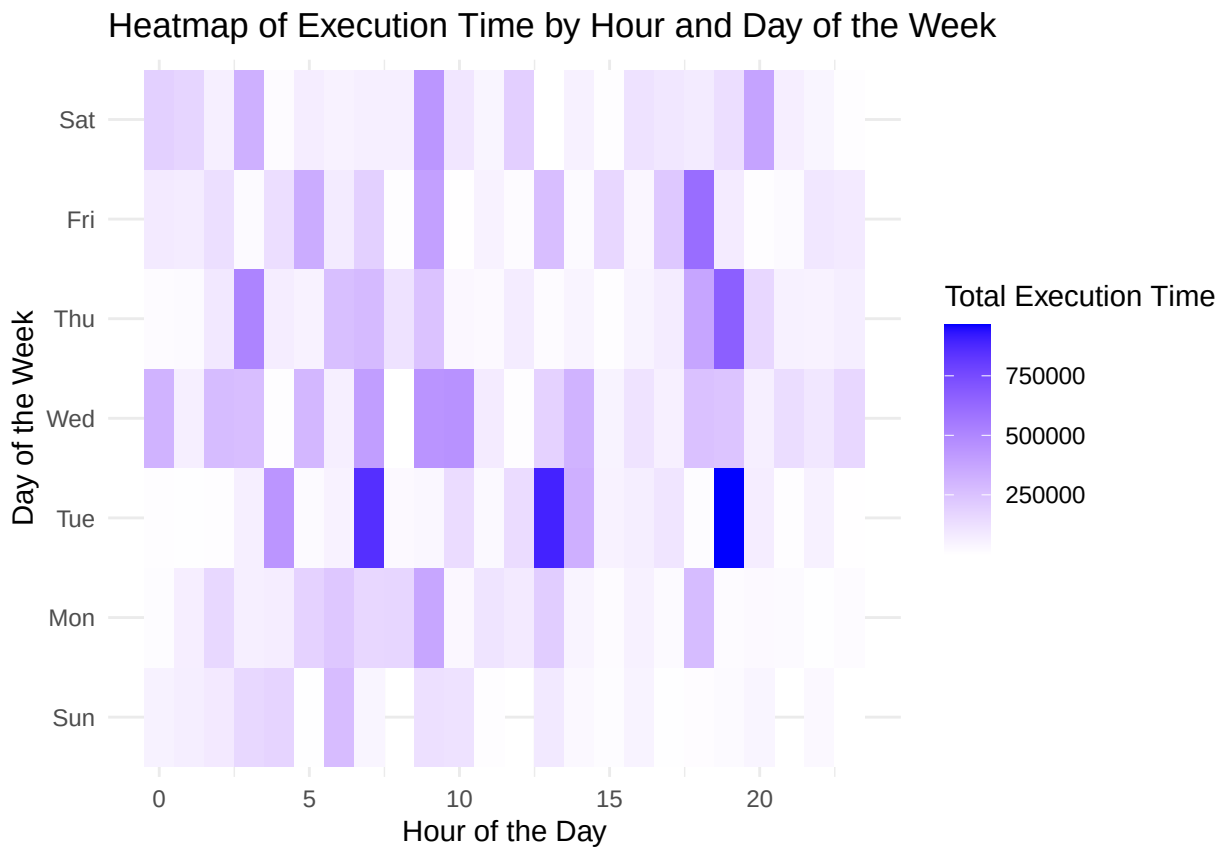
```
## <Training/Validation/Testing/Total>
## <2640/880/881/4401>
```

Q1: How does the computational workload on the Cori supercomputer exhibit distinct patterns or variations throughout the week?

```r
train_data1 <- train_data

# Extracting the hour and day of the week from the start time
train_data1$hour <- hour(train_data1$start)
train_data1$day_of_week <- wday(train_data1$start, label = TRUE)

# Calculating average total execution time
total_executiontime <- train_data1 |>
  group_by(day_of_week, appname) |>
  summarise(total_executiontime = sum(executiontime))

ggplot(total_executiontime, aes(x = day_of_week, y = total_executiontime, fill = appname)) +
  geom_bar(stat = "identity") +
  theme_minimal() +
  labs(x = "Day of the Week", y = "Total Execution Time", title = "Total Execution Time by Day of the W
```




```r
# Calculating the total execution time for each hour of each day of the week
total_executiontime_hourly <- train_data1 |>
  group_by(day_of_week, hour) |>
  summarise(total_executiontime = sum(executiontime, na.rm = TRUE))

# Creating a heatmap
ggplot(total_executiontime_hourly, aes(x = hour, y = day_of_week, fill = total_executiontime)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "blue") +
  theme_minimal() +
  labs(x = "Hour of the Day", y = "Day of the Week", fill = "Total Execution Time", title = "Heatmap of
```



INTERPRETATION:

Yes, the computational workload on the Cori supercomputer exhibit distinct patterns or variations throughout the week

Bar chart:

Weekdays exhibit higher job execution times, gradually decreasing towards weekends before rising again on Mondays, suggesting increased computational load during weekdays compared to weekends.

Note:

But we also need to look into a thing that more execution time means more computational load but we can see a trend

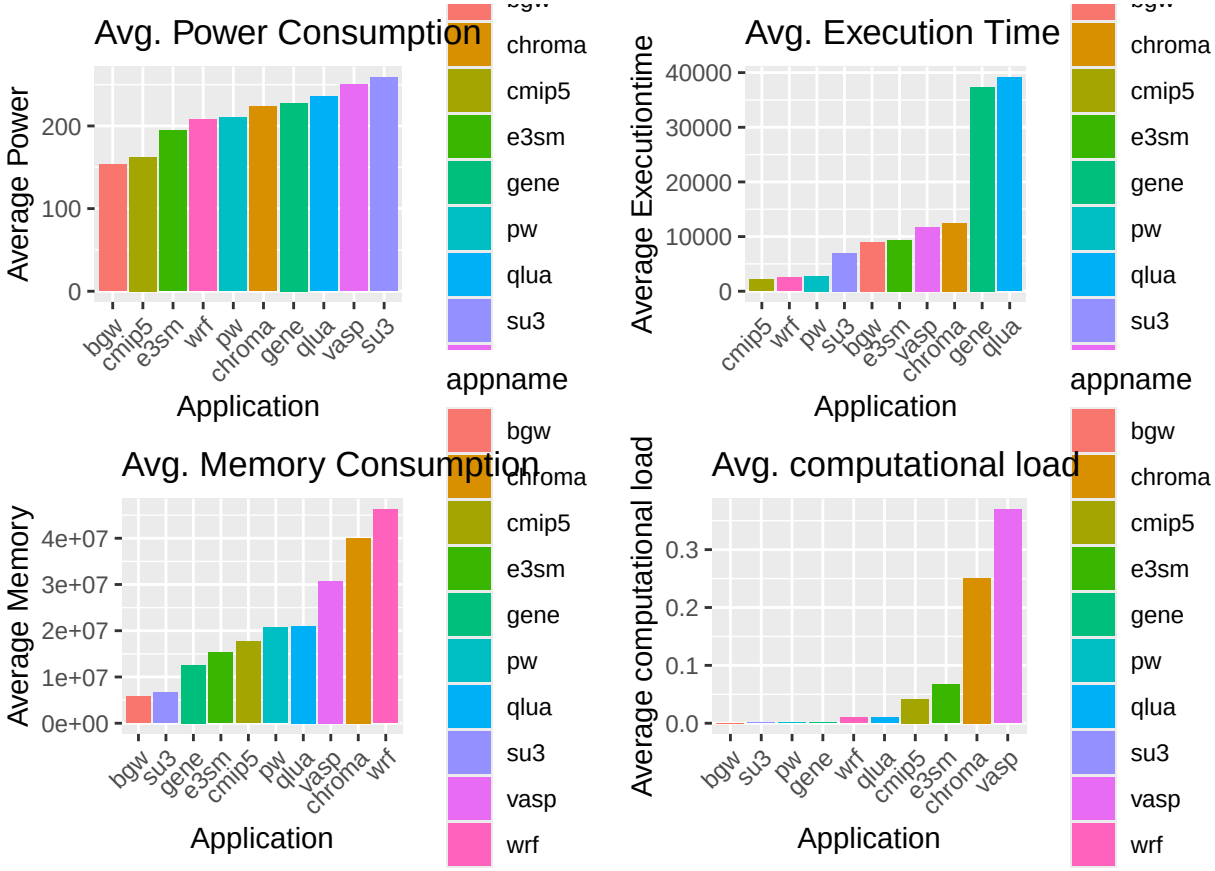
Heat map:

The heat map tells us Peak computational load typically occurs between 6am to 7pm.

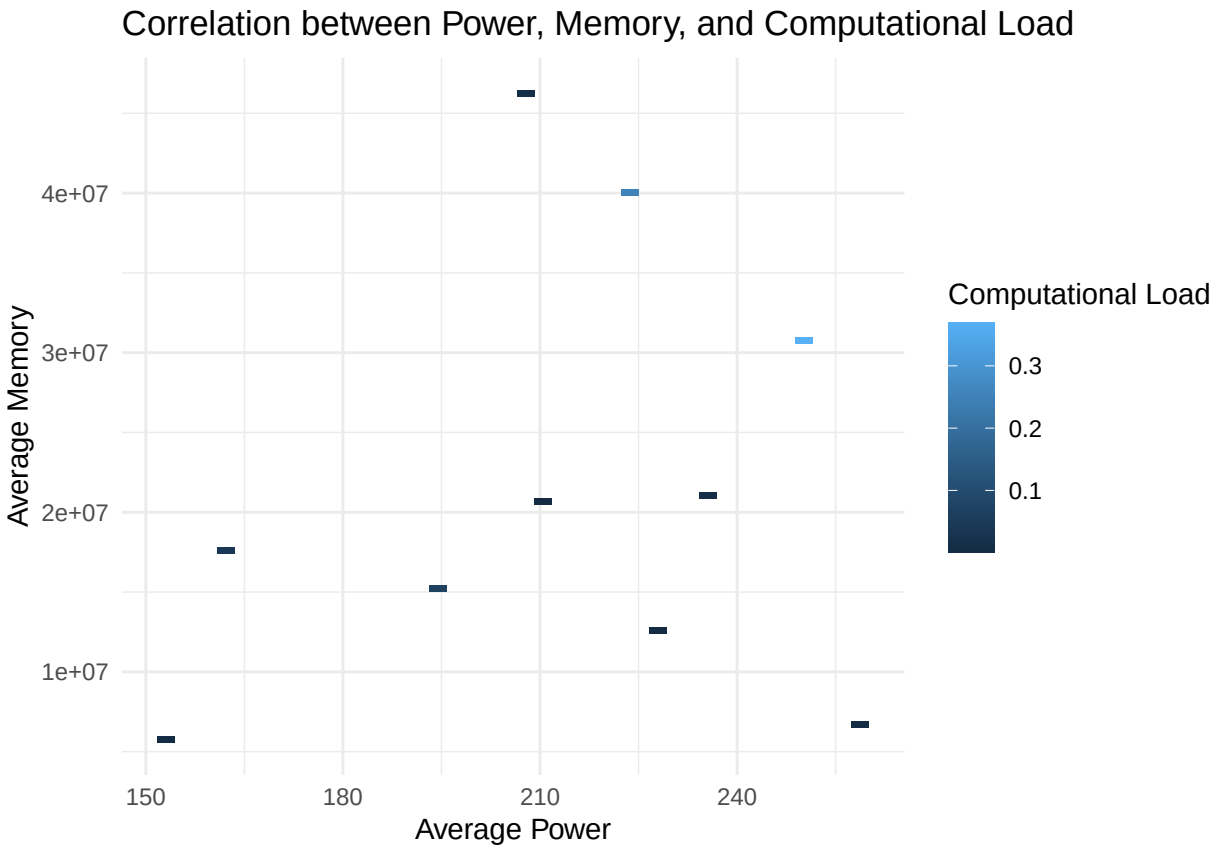
Q2: How does the type of application influence resource consumption in terms of power, memory usage, and computational load in large scale computing environments like cori?

```
# Calculating average resource consumption
grouped_data <- train_data |>
  filter(memory >= 0) |>
  group_by(appname) |>
  summarise(
    average_executiontime = mean(executiontime, na.rm = TRUE),
    average_power = mean(power, na.rm = TRUE),
    average_memory = mean(memory, na.rm = TRUE),
```

```
    average_tasks_per_cpu_per_second = sum(taskspercpu * numcpus / executiontime, na.rm = TRUE) / n()
  )

# Plotting Average Resource Consumptions
plot1 <- ggplot(grouped_data, aes(x = reorder(appname, average_power), y = average_power, fill = appname
  geom_bar(stat = "identity") +
  labs(title = "Avg. Power Consumption",
       x = "Application", y = "Average Power") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

plot2 <- ggplot(grouped_data, aes(x = reorder(appname, average_executiontime), y = average_executiontime
  geom_bar(stat = "identity") +
  labs(title = "Avg. Execution Time",
       x = "Application", y = "Average Executiontime") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

plot3 <- ggplot(grouped_data, aes(x = reorder(appname, average_memory), y = average_memory, fill = appna
  geom_bar(stat = "identity") +
  labs(title = "Avg. Memory Consumption ",
       x = "Application", y = "Average Memory") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

plot4 <- ggplot(grouped_data, aes(x = reorder(appname, average_tasks_per_cpu_per_second), y = average_ta
  geom_bar(stat = "identity") +
  labs(title = "Avg. computational load ",
       x = "Application", y = "Average computational load") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Combine plots using cowplot
combined_plot <- plot_grid(plot1, plot2, plot3, plot4, ncol = 2)

combined_plot
```




```r
# Heatmap for Correlation
ggplot(grouped_data, aes(x = average_power, y = average_memory, fill = average_tasks_per_cpu_per_second
  geom_tile() +
  labs(title = "Correlation between Power, Memory, and Computational Load",
       x = "Average Power", y = "Average Memory", fill = "Computational Load") +
  theme_minimal()
```



INTERPRETATION:

Applications with higher power and memory consumption typically correlate with increased computational loads, as computationally intensive tasks often demand more power and memory.

Bar plots:

The applications chroma, gene, qlu a, vasp, and su 3 have more than 200 average power consumption.

The applications chroma, gene, qlu a, and vasp have higher average execution time.

The applications chroma, qlu a, vasp, and wrf have higher average memory consumption.

The applications chroma, qlu a, vasp, cmip 5, and e3sm also exhibit high computational load.

Applications like chroma, gene, qlu a, and vasp significantly influence resource consumption in NERSC Cori, highlighting their major impact on computational load.

Heat map: Positive correlation between power consumption and computational load

Weak correlation between memory usage and computational load, the shades remain relatively consistent across different memory usage levels.

Weak correlation between power consumption and memory usage, the shades are scattered without a distinct trend.

In Cori supercomputer, computational load strongly correlates with power consumption rather than memory usage. Tasks with higher computational loads consume more power, aligning with expectations.

However, Correlation between memory usage and power consumption, as well as computational load, seems weaker.

Note: These observations are based solely on the information presented in this specific heatmap. Further analysis, considering additional factors or variables, may be necessary.

From the interpretation it suggest that the type of application influence resource consumption in terms of power, memory usage, and computational load in large scale computing environments like cori.

Q3: What is the optimal predictive model for forecasting power usage of future jobs and applications on the NERSC Cori supercomputer?

LM MODEL

```
#numnodes and numcpus are multicollinear so considering only "numnodes" variable

selected_features <- c("numnodes", "numtasks", "executiontime", "memory", "taskspercpu", "taskspernode")
target <- "power"


# Cross Validation

num_folds <- 10
num_repeats <- 3


ctrl <- trainControl(method = "repeatedcv",
                     number = num_folds,
                     repeats = num_repeats,
                     search = "grid")

# Formula for the lm model
formula <- power ~ numnodes + numtasks + executiontime + memory + taskspercpu + taskspernode

# Train the model using the selected features
lm_model <- train(formula,
                  data = train_data,
                  method = "lm",
                  trControl = ctrl)



# Calculating mean Cross Validated MSE
lm_cv_mse <- mean(lm_model$results$RMSE)


print(paste("Mean Cross Validated RMSE (linear Regression):", lm_cv_mse))
```

```
## [1] "Mean Cross Validated RMSE (linear Regression): 35.5553456134124"
```

```
# Model evaluation using validation_data

lm_predictions <- predict(lm_model, newdata = validation_data[, selected_features])

#rsquared <- summary(lm_model)$r.squared
```

```r
lm_mse <- mean((lm_predictions - validation_data$power)^2)
lm_rmse <- sqrt(lm_mse)
lm_mae <- mean(abs(lm_predictions - validation_data$power))

#Evaluation Metrics

#print(paste("R-squared (R²):", rsquared))
print(paste("Mean Squared Error MSE (linear Regression):", lm_mse))
```

```
## [1] "Mean Squared Error MSE (linear Regression): 1207.90014784776"
```

```r
print(paste("Root Mean Squared Error RMSE (linear Regression):", lm_rmse))
```

```
## [1] "Root Mean Squared Error RMSE (linear Regression): 34.7548579028567"
```

```r
print(paste("Mean Absolute Error MAE (linear Regression):", lm_mae))
```

```
## [1] "Mean Absolute Error MAE (linear Regression): 28.4387620461457"
```

```r
# Make predictions for future jobs and applications
#future_predictions <- predict(lm_model, newdata = test_data[, selected_features])
```

INTERPRETATION:

Lower Mean Cross-Validated RMSE, MSE/RMSE, and MAE signify better model performance and a closer fit to the data.

But the linear model is unable to capture nonlinear relationships in the data and suggests to exploration of polynomial regression that can better capture the underlying non linear relationship.

POLYNOMIAL REGRESSION MODEL

```r
selected_features <- c("numnodes", "numtasks", "executiontime", "memory", "taskspercpu", "tasksperno
target <- "power"

# Formula for  polynomial Regression
formula <- power ~ numnodes + numtasks + executiontime + memory + taskspercpu + tasksperno
  I(numnodes^2) + I(numtasks^2) + I(executiontime^2) + I(memory^2) + I(taskspercpu^2) + I(tasksperno

# Cross Validation
num_folds <- 10
num_repeats <- 3
ctrl <- trainControl(method = "repeatedcv",
                     number = num_folds,
                     repeats = num_repeats,
                     search = "grid")

# Training polynomial regression model using train() function
set.seed(123)
```

```r
poly_lm_model <- train(formula,
                       data = train_data,
                       method = "lm",
                       trControl = ctrl)

# Calculating mean Cross Validated RMSE
poly_cv_rmse <- mean(poly_lm_model$results$RMSE)

print(paste("Mean Cross Validated RMSE (Polynomial Regression):", poly_cv_rmse))
```

```
## [1] "Mean Cross Validated RMSE (Polynomial Regression): 32.8377831397941"
```

```r
# Model evaluation and Prediction with polynomial regression model using validation data

poly_predictions <- predict(poly_lm_model, newdata = validation_data[,selected_features])

# Calculating Evaluation metrics

#poly_rsquared <- R2(poly_predictions, validation_data[[target]])
poly_mse <- mse(poly_predictions, validation_data[[target]])
poly_rmse <- rmse(poly_predictions, validation_data[[target]])
poly_mae <- mae(poly_predictions, validation_data[[target]])



#print(paste("R-squared (R²) - Polynomial Regression:", poly_rsquared))
print(paste("Mean Squared Error MSE (Polynomial Regression):", poly_mse))
```

```
## [1] "Mean Squared Error MSE (Polynomial Regression): 988.488227526519"
```

```r
print(paste("Root Mean Squared Error (RMSE Polynomial Regression):", poly_rmse))
```

```
## [1] "Root Mean Squared Error (RMSE Polynomial Regression): 31.4402326251973"
```

```r
print(paste("Mean Absolute Error MAE (Polynomial Regression):", poly_mae))
```

```
## [1] "Mean Absolute Error MAE (Polynomial Regression): 24.2873735586808"
```

INTERPRETATION:

The drastic decrease in Mean Cross Validated RMSE from linear regression = 1264.18 to polynomial regression =32.84 indicates a substantial improvement in the predictive performance of the polynomial regression model suggesting that the polynomial regression model captures the underlying relationship between the predictor variables and the target variable more effectively compared to the linear regression model.

Note: There is risk of overfitting in polynomial models due to their tendency to memorize training data, which suggests that exploring the random forest model can provide improved generalization to unseen data.

RANDOM FOREST

```r
train_data <- as.data.frame(train_data)
validation_data <- as.data.frame(validation_data)


selected_features <- c("numnodes", "numtasks", "executiontime", "memory", "taskspercpu", "taskspernode")
target <- "power"

# Cross Validation
num_folds <- 10
num_repeats <- 3
ctrl <- trainControl(method = "repeatedcv",
                     number = num_folds,
                     repeats = num_repeats,
                     search = "grid")

# Training random forest regression model
set.seed(123)
rf_model <- train(train_data[, selected_features],
                  train_data[[target]],
                  method = "rf",
                  trControl = ctrl)

# Calculating mean Cross Validated RMSE
rf_cv_rmse <- mean(rf_model$results$RMSE)


print(paste("Mean Cross Validated RMSE (Random Forest Regression):", rf_cv_rmse))
```

```
## [1] "Mean Cross Validated RMSE (Random Forest Regression): 11.8226905138733"
```

```r
# Evaluating random forest model using validation data
rf_predictions <- predict(rf_model, newdata = validation_data[, selected_features])

# Calculate evaluation metrics

#rf_rsquared <- R2(rf_predictions, validation_data[[target]])
rf_mse <- mse(rf_predictions, validation_data[[target]])
rf_rmse <- rmse(rf_predictions, validation_data[[target]])
rf_mae <- mae(rf_predictions, validation_data[[target]])


#print(paste("R-squared (R²) - Random Forest Regression:", rf_rsquared))
print(paste("Mean Squared Error MSE (Random Forest Regression):", rf_mse))
```

```
## [1] "Mean Squared Error MSE (Random Forest Regression): 140.659055096961"
```

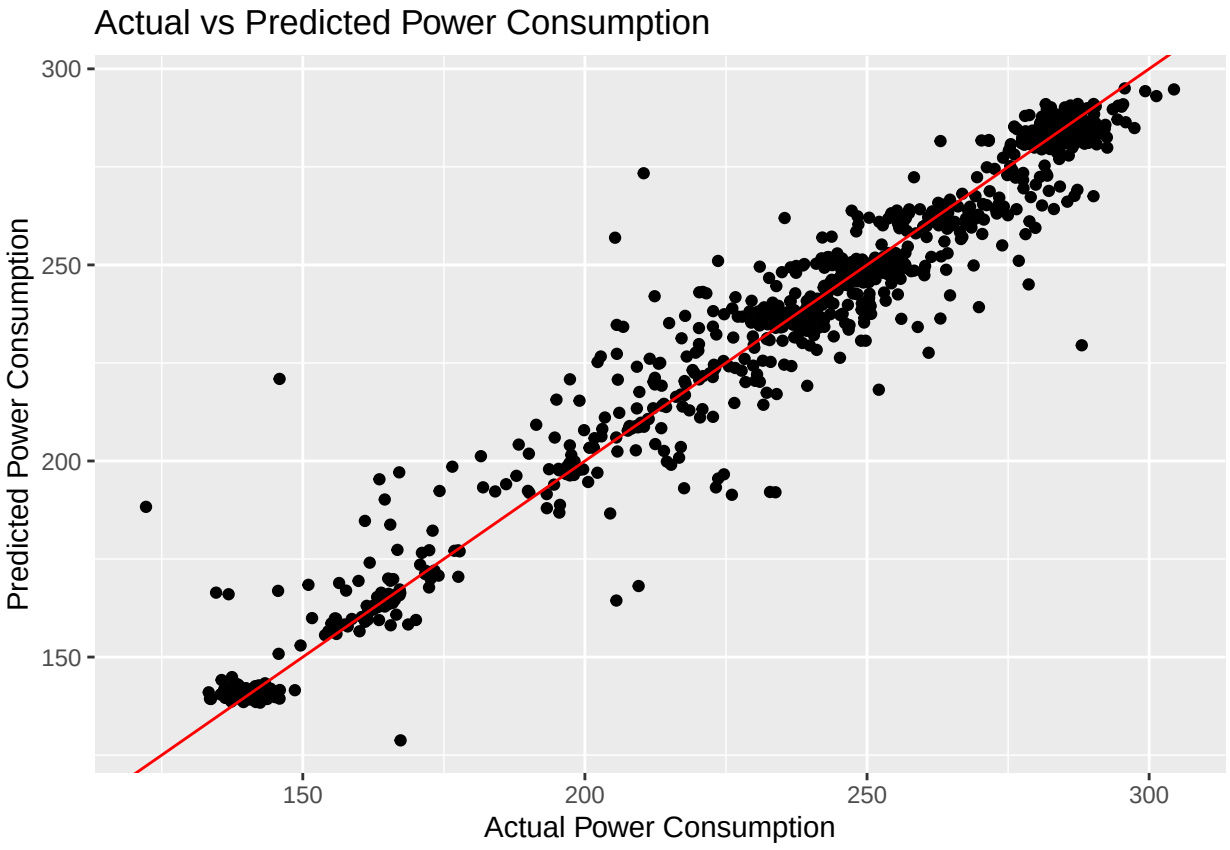
```r
print(paste("Root Mean Squared Error RMSE (Random Forest Regression):", rf_rmse))
```

```
## [1] "Root Mean Squared Error RMSE (Random Forest Regression): 11.8599770276743"
```

```r
print(paste("Mean Absolute Error MAE (Random Forest Regression):", rf_mae))
```

```
## [1] "Mean Absolute Error MAE (Random Forest Regression): 6.6688104824263"
```

```r
future_predictions <- predict(rf_model, newdata = test_data[, selected_features])

#future_predictions


comparison <- data.frame(Actual = test_data$power, Predicted = future_predictions)


# Scatter plot to visualize the predictions

# Red line represents perfect prediction
ggplot(comparison, aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red") +
  ggtitle("Actual vs Predicted Power Consumption") +
  xlab("Actual Power Consumption") +
  ylab("Predicted Power Consumption")
```



INTERPRETAIOIN:

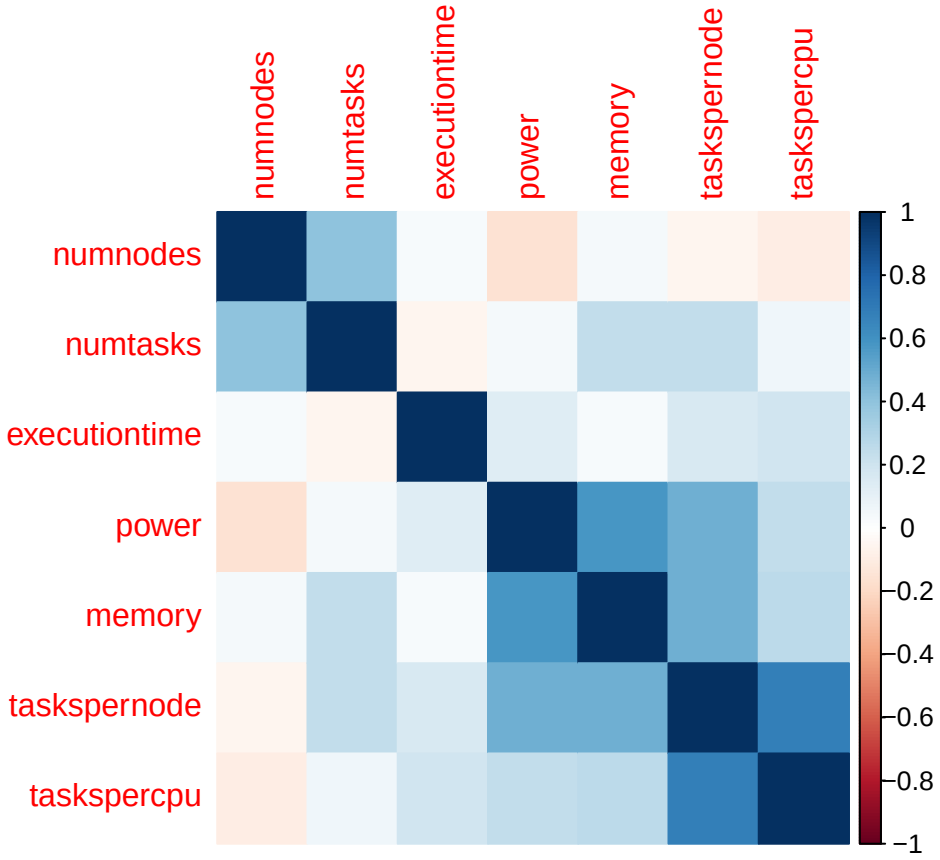
From above metrics the Random Forest model, on average, makes smaller errors in its predictions.

Based on the above metrics, it can be inferred that the random forest regression model outperforms the polynomial regression model, linear model in terms of prediction accuracy and generalization to new data. The lower values of RMSE, MSE, RMSE, and MAE for the random forest model suggest that it provides more reliable predictions of power usage for future jobs and applications on the NERSC Cori supercomputer.

This way Random forest potentially capturing more complex and non linear relationships between features and power consumption, which the linear and polynomial models have failed to identify the underlying relationship.

Q4: What are the correlations between power consumption, memory usage, and other variables on the Cori cluster, and how can these correlations inform strategies to optimize energy efficiency?

```
selected_variables <- c("numnodes","numtasks", "executiontime","power", "memory","taskspernode","tasksp

# Correlation matrix calculation
correlation_matrix <- cor(train_data[, selected_variables])

# Visualizing correlation
corrplot(correlation_matrix, method = "color")
```



INTERPRETATION:

- **Strong positive correlation** (taskspernode) & (taskspercpu) which is an expected relationship, as tasks are typically distributed across nodes and CPUs based on the available resources.
- **Moderate positive correlation** (power) & (memory) suggests that applications or workloads with higher memory requirements tend to consume more power, which is an important consideration for resource allocation and energy efficiency strategies.

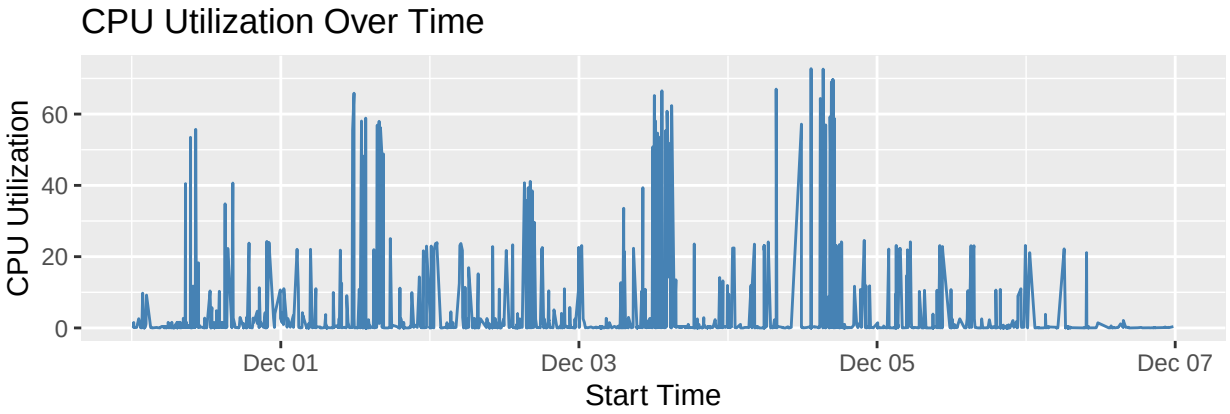
- **Moderate positive correlation** of 0.408 (numnodes) & (numtasks). due to increase in nodes typically handle larger computational workloads.
- **Weak positive correlation** (numnodes) & (executiontime) which implies that increasing the number of nodes may not necessarily lead to a significant reduction in execution time, as other factors also play a role.
- **Weak positive correlation** (executiontime) & (power) which is an expected relationship, as longer execution times generally lead to higher power consumption.
- **weak negative correlation** (numnodes) & (power).This counterintuitive relationship suggests that simply increasing the number of nodes may not result in a proportional increase in power consumption, possibly due to efficient power management strategies or workload distribution across nodes.
- **Weak negative correlation** (numtasks) & (executiontime)implies that increasing the number of tasks may slightly reduce the execution time, possibly due to parallel processing or workload distribution.

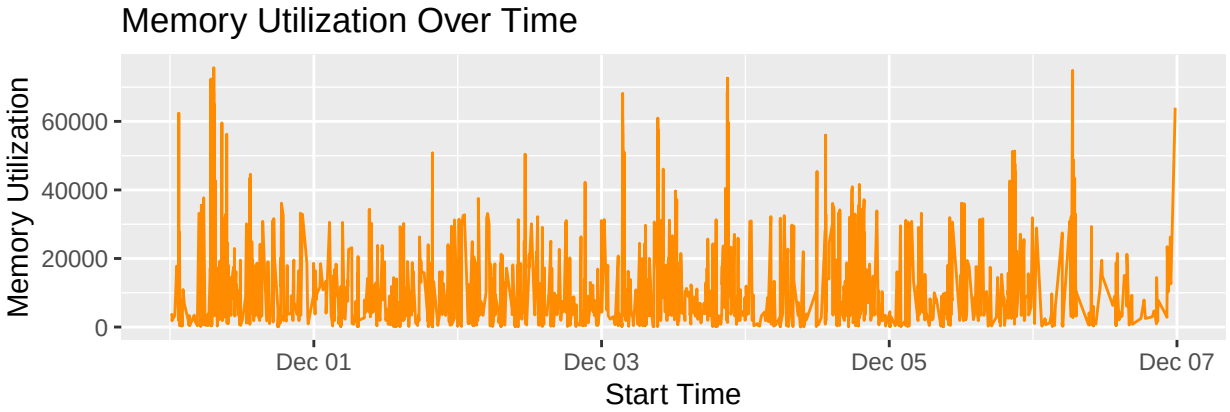
THEREFORE The moderate positive correlation between power and memory suggests that optimizing memory usage could lead to potential energy savings. And the weak negative correlation between numnodes and power implies that simply increasing the number of nodes may not be an efficient strategy for power management.

NOTE: These correlations do not necessarily suggests exact reflections, domain expertise and more data may be necessary to develop effective resource allocation and energy optimization strategies for the Cori supercomputer.

Q5: Are there any particular areas in any need of strategies for improvement in optimal resource utilisation by identifying high utilization or intensive periods on NERSC Cori?

```
train_data6 <- train_data

# Calculating resource utilization metrics
train_data6 <- train_data6 |>
  mutate(cpu_utilization = numcpus / executiontime,
         memory_utilization = memory / executiontime)

train_data6 <- train_data6 |>
  filter(memory_utilization >= 0)

plot_cpu <- ggplot(train_data6, aes(x = start, y = cpu_utilization)) +
  geom_line(color = "steelblue") +
  labs(x = "Start Time", y = "CPU Utilization",
       title = "CPU Utilization Over Time")

plot_memory <- ggplot(train_data6, aes(x = start, y = memory_utilization)) +
  geom_line(color = "darkorange") +
  labs(x = "Start Time", y = "Memory Utilization",
       title = "Memory Utilization Over Time")

# Arranging the plots in a sigle column
plot_grid(plot_cpu, plot_memory, ncol = 1)
```





INTERPRETATION:

CPU utilization: Peaks in the graph indicate periods of high CPU utilization, suggesting intensive computational workloads or high CPU demand and those are:

- December 1st (early hours)
- December 3rd (mid-day)
- December 5th (mid-day and evening)
- December 6th (mid-day and evening)

Memory Utilization: The spikes in the memory utilization plot indicate periods of high memory usage, highlighting significant high memory utilization periods and those are:

- December 1st (early hours)
- December 3rd (mid-day)
- December 5th (mid-day and evening)
- December 6th (mid-day and evening)
- December 7th (early hours)

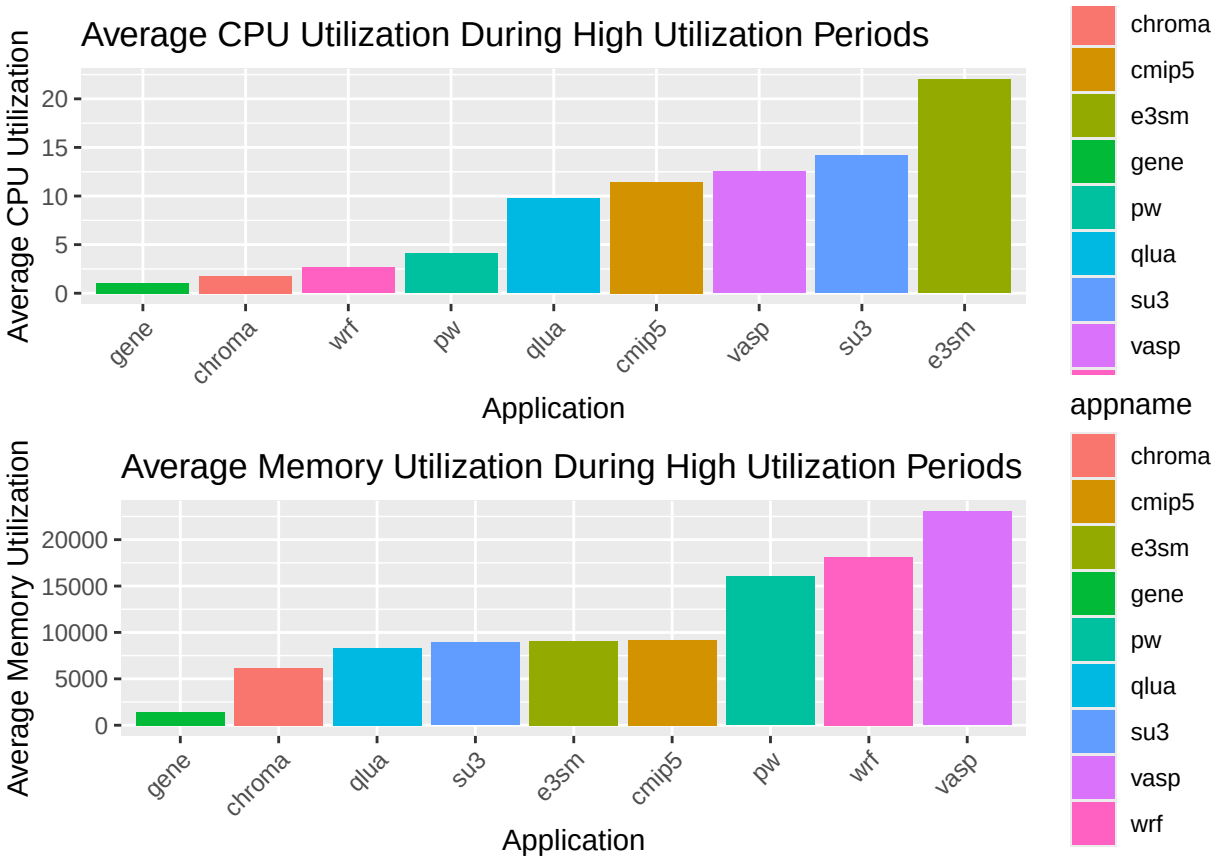
The high CPU and memory utilization periods often coincide which suggest that resource intensive applications are consuming both CPU and memory demands simultaneously.

This way we can develop strategies for optimal resource utilisation on above dates by:

- Analyzing the applications running during these periods to understand their resource requirements and potential optimization opportunities.

- Implementing workload scheduling and load balancing strategies.
- Otimizing resource intensive applications that contribute significantly to high utilization periods.

```
# Calculating resource utilization metrics
train_data6 <- train_data |>
  mutate(cpu_utilization = numcpus / executiontime,
         memory_utilization = memory / executiontime)

# Filtering out negative memory utilization (if any)
train_data_filtered <- train_data6 |>
  filter(memory_utilization >= 0)

# Defining a threshold for both CPU and memory utilization are 60% or more
high_utilization_threshold <- 0.6

# Identifying periods with high CPU and memory utilization
high_utilization_periods <- train_data_filtered |>
  filter(cpu_utilization >= high_utilization_threshold & memory_utilization >= high_utilization_thresho

# Calculating average resource utilization
application_usage <- high_utilization_periods |>
  group_by(appname) |>
  summarize(avg_cpu_utilization = mean(cpu_utilization),
            avg_memory_utilization = mean(memory_utilization))

# Visualising average resource utilization
plot_cpu <- ggplot(application_usage, aes(x = reorder(appname, avg_cpu_utilization), y = avg_cpu_utiliz
  geom_bar(stat = "identity") +
  labs(title = "Average CPU Utilization During High Utilization Periods", x = "Application", y = "Avera
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

plot_memory <- ggplot(application_usage, aes(x = reorder(appname, avg_memory_utilization), y = avg_memo
  geom_bar(stat = "identity") +
  labs(title = "Average Memory Utilization During High Utilization Periods", x = "Application", y = "Av
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Arranging the plots in a single column
plot_grid(plot_cpu, plot_memory, ncol = 1)
```



The top performing applications during high utilisation periods are vasp,su 3,pw,wrf,qlua, e3sm

STRATEGIES:

- *Application optimization:* Applications with consistently high resource utilization during intensive periods could benefit from code optimizations, algorithmic improvements, or alternative approaches to reduce their resource usage.
- *Resource allocation and job scheduling:* The Applications with high resource demands could be allocated dedicated resource pools or scheduled in a manner that minimizes resource contention and improves overall system performance.
- *Load balancing:* Implementing load balancing techniques to distribute resource intensive workloads across multiple nodes or clusters can help lessen the periods of high resource utilization on the Cori supercomputer.
- *Capacity planning:* Identifying recurring patterns of high resource utilization can inform capacity planning decisions, such as hardware upgrades, resource allocation & distribution , or the deployment of additional computational resources.

Conclusion [15 points]

Does the analysis answer the research questions?

The analysis addresses research questions by examining variations in computational loads throughout the week on Cori, identifying optimal regression models for precise predictions, understanding data relationships for future power consumption forecasts, uncovering correlations and multicollinearity, and developing

strategies based on resource intensive periods. Additionally, it identifies applications during the resource intensive periods influencing resource consumption in terms of power, memory usage, and computational load in large-scale computing environments like Cori. Therefore, the analysis answers all the research questions with best model for predicting.

Discuss the scope and generalizability of the analysis.

The analysis comprehensively explores computational load on Cori, encompassing descriptive statistics, predictive modeling, correlation analysis, and resource utilization strategy development to understand patterns, predict power usage, identify correlations, and optimize resource utilization.

The generalizability of the analysis is limited due to the specific nature of the dataset. The data is from NERSC's Cori supercomputer, and the findings might not be directly applicable to other supercomputers or computing environments. However, the methodologies and approaches used in this analysis could be adapted and applied to similar datasets or environments.

Example, Predictive modeling, correlation analysis, and resource utilization strategies have broader applicability beyond Cori, offering insights for other high-performance computing environments.

However, it's important to note that each computing environment has its unique characteristics and challenges. Therefore, while the analysis provides valuable insights and methodologies, it would be necessary to consider the specific context and characteristics of the other environments when applying these findings.

In terms of improvement, the analysis could benefit from a larger and more diverse dataset. This would allow for a more comprehensive understanding of the system and could potentially enhance the accuracy and reliability of the predictive models. Additionally, addressing the instances of negative memory consumption and checking against multicollinearity could also improve the quality of the analysis.

Overall, while the analysis provides valuable insights into the computational load on the Cori supercomputer, its generalizability to other systems or environments would require careful consideration and potential adjustments.

Discuss potential limitations and possibilities for improvement.

Limitations: - The dataset spans only one week, limiting the depth of analysis for a complex system like Cori. - Multicollinearity poses a challenge, potentially skewing results. - There are few observations with negative memory consumption that could be because of erroneous data or unaccounted/other factors.

Possibilities for Improvement: - Utilizing larger datasets, such as monthly or yearly data, would offer deeper insights into trends and patterns on Cori. - Consistent check against multicollinearity is crucial for more accurate results. - Addressing the above instances of negative memory consumption requires specialized domain knowledge and expertise about NERSC Cori.