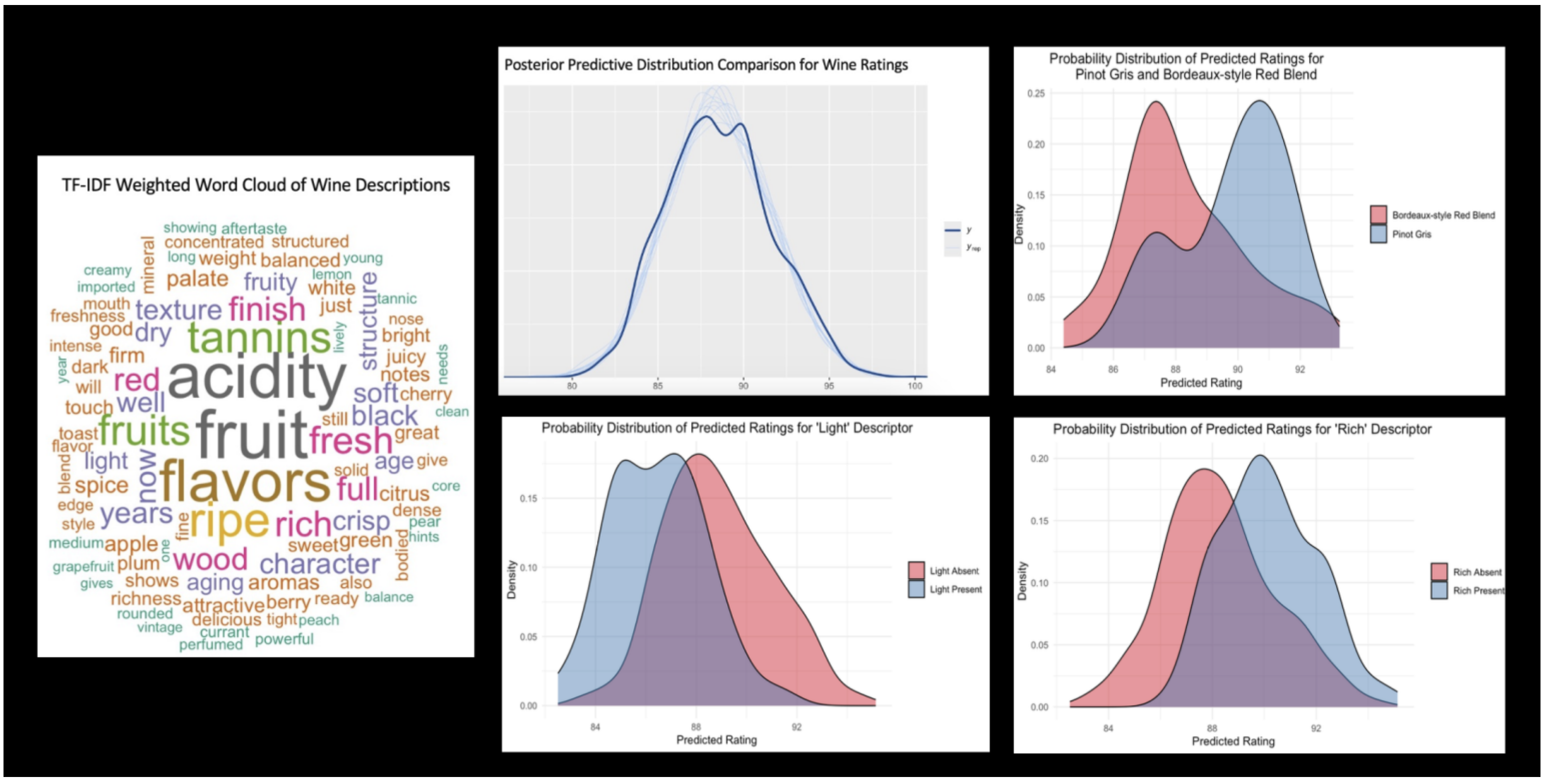
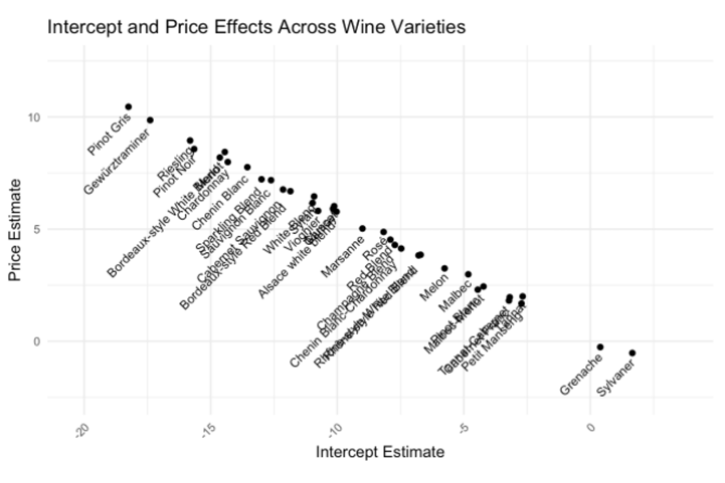
CS7DS3 Main Assignment

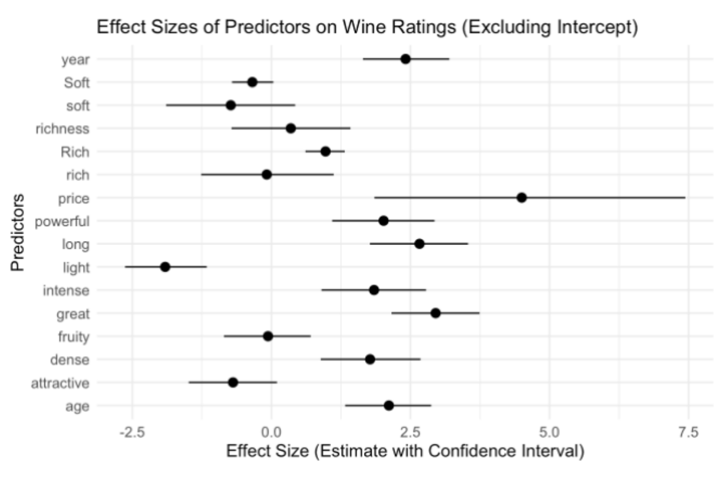
Nachiketh J (23337215) **03.05.2024**

Executive Summary

This report presents a detailed statistical analysis of a dataset comprising wine reviews, focusing on ratings of various French wine varieties. Our primary objective was to develop a predictive model to estimate wine ratings and to classify wines as "superior" based on a rating threshold of 90 points.







x

Using a Bayesian regression model, I aimed to uncover key factors influencing wine ratings and to assess the variability in ratings across different wine varieties.
Key Findings from the Analysis Include:

- **Price and Age as Strong Predictors**: The model identified price and age as significant predictors of wine ratings, where higher prices and greater age generally correspond to higher ratings, suggesting that consumers associate cost and maturity with quality.

- **Variety-Specific Effects**: The analysis revealed notable differences in base ratings and price sensitivity among wine varieties. Some varieties demonstrated higher baseline ratings irrespective of price, while others were significantly more sensitive to price changes.

- **Predictive Performance**: The Bayesian model achieved a good balance between complexity and interpretability, with an **R-squared of 0.629** and a **root mean square error (RMSE) of 1.913**. The accuracy for the classification from the points prediction is 79%. These metrics indicate strong predictive capability relative to the variability in the data.

- **Text Analysis Insights**: A weighted word cloud of wine descriptions highlighted terms most associated with higher ratings, such as "balanced," "rich," and "fruity." This suggests that certain descriptors in wine reviews positively influence consumer perceptions and ratings.

Central Figure: The central figure consolidates these insights and features:

- TF-IDF Weighted Word Cloud of Wine Descriptions: Displays the most influential words from wine descriptions, sized by their TF-IDF score to emphasize their importance.

- Posterior Predictive Distribution Comparison for Wine Ratings: Compares observed wine ratings with those predicted by the model, visualizing the model's accuracy and fit.

- Probability Distribution of Predicted Ratings for 'Light' Descriptor: Shows the impact of the descriptor "light" on wine ratings, with separate distributions for wines with and without this descriptor.

- Probability Distribution of Predicted Ratings for Pinot Gris and Bordeaux-style Red Blend: Illustrates the predicted rating differences between Pinot Gris and Bordeaux-style Red Blend, highlighting varietal distinctions.

- Probability Distribution of Predicted Ratings for 'Rich' Descriptor: Demonstrates how the descriptor "rich" influences predicted wine ratings, comparing distributions for presence versus absence of this term.

- Intercept and Price Effects Across Wine Varieties: Visualizes the relationship between intercept estimates and price effects for various wine varieties, indicating how base ratings and price sensitivity differ across types.

- Effect Sizes of Predictors on Wine Ratings (Excluding Intercept): Depicts the estimated effect sizes and confidence intervals for various predictors of wine ratings, highlighting the magnitude and uncertainty of each predictor's impact on the ratings.

Introduction

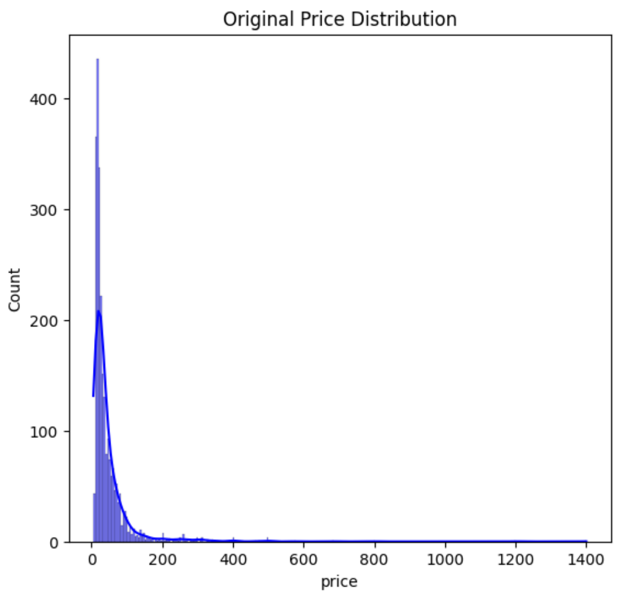
- The intricacies of wine quality and its perception have long fascinated both connoisseurs and casual consumers alike. Expert and enthusiast-provided wine ratings are crucial in the industry, influencing purchasing decisions, pricing, and market trends.
- In this context, our report aims to find the factors that significantly impact the ratings of French wines, using a robust dataset of wine reviews. The primary goal of this analysis is twofold: to predict the numerical ratings of wines and there using it to classify wines as "superior" based on a threshold of 90 points.
- Our analysis utilizes a dataset rich in variables, from fundamental attributes like price and variety to detailed data from expert reviews. Using a Bayesian regression model, I aim to accurately predict wine ratings and identify the key attributes that correlate with higher ratings, offering a detailed analysis of the factors influencing wine quality.
- Additionally, our analysis investigates the consistency and reliability of wine ratings across different varieties, exploring whether certain wines are rated higher or specific descriptors influence ratings. These insights could significantly impact marketing strategies and consumer expectations in the wine market.
- Using advanced statistical methods, this report will provide actionable insights that clarify the factors influencing wine ratings. Readers will gain a better understanding of the dynamics affecting wine quality assessments and how to effectively predict them.

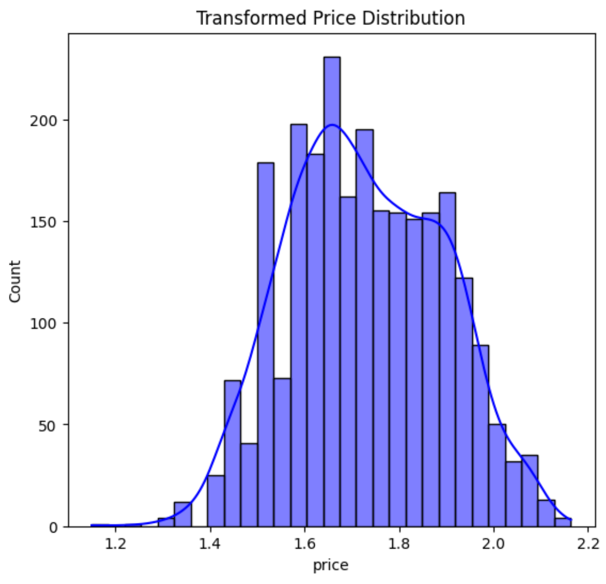
Data Description

- Data Source:
  The dataset used in our analysis originates from a comprehensive collection of wine reviews focusing on French wines, sourced from a public dataset available on Kaggle. This dataset was chosen due to its extensive detail on wine ratings and the variety of variables it includes.

- Variable Descriptions:
  The primary variables in our dataset include:
  - **Points**: Wine ratings on a scale of 0 to 100, with scores of 90 or above indicating "superior" wines.
  - **Price**: The retail price of each wine in euros.
  - **Variety**: Type of grape used in the wine, which influences its taste and rating.
  - **Description**: Free-text reviews from wine critics that describe sensory attributes such as aroma, body, and flavour.

- Data Pre-processing:
  Several pre-processing steps were necessary to prepare the dataset for analysis:
  - **Text Processing**:
    - I applied natural language processing techniques to the 'Description' text. This involved converting text to lowercase, removing punctuation and non-alphabetic characters, tokenizing, and filtering out stop words and custom-defined irrelevant words like 'year' and 'wine'.
    - I then used lemmatization based on parts of speech to derive the root forms of words, ensuring the text data was standardized for analysis.
  - **TF-IDF Vectorization**:
    - Post text pre-processing, a TF-IDF Vectorizer was employed to transform the lemmatized text into a numerical format by extracting the 100 most significant features. This helped in quantifying the impact of different words on wine ratings.





  - **Price Transformation**:
    - Given the skewed distribution of the 'Price' variable, I performed a logarithmic transformation to normalize its distribution. For prices not fitting a normal distribution, a Box-Cox transformation was applied, improving the suitability of this variable for regression analysis.

  - **Feature Selection**:
    - We calculated correlation coefficients for numeric variables with respect to 'Points' to identify which features strongly correlate with wine ratings. Variables with a correlation above a 0.15 threshold were retained for further analysis.

- **Data Cleaning**:
    - Initial steps included removing duplicate entries and handling missing values through imputation or exclusion, based on their significance and the proportion of missing data.

Dataset Finalization: After pre-processing, the dataset was reduced to essential features with potential predictive power regarding wine ratings. The final cleaned and transformed dataset was saved as 'final_df.csv', ready for the subsequent modelling phase.

Analysis

Methodological Details:

Our analytical approach involved both statistical modelling and machine learning techniques to predict wine ratings and identify key attributes influencing these ratings. Initially, I employed a Bayesian regression model, chosen for its flexibility in handling complex, non-linear relationships and its ability to incorporate prior knowledge or assumptions into the model.

- Bayesian Regression: This model was particularly suited for our dataset with its mixed data types and varying scales of measurement. Bayesian methods allowed us to estimate a posterior distribution for each parameter in our model, providing a comprehensive view of parameter uncertainty.

Model Building: The process of building our model involved several iterative steps:

- Initial Simple Model: I started with a basic model that included only the most straightforward variables such as price and binary features given in the data without making use of description column. This initial model helped establish a baseline for comparison.

- Feature Engineering and Selection: Based on the TF-IDF analysis of the description texts, I added more complex features related to the textual data, which we hypothesized might have a significant impact on ratings. Variables were selected based on their correlation with the target variable (points) and practical relevance.
- Complex Model Building: Incorporating more variables and interactions, I refined our model iteratively, assessing each version's performance using cross-validation techniques and adjusting our feature set and model parameters based on the results.

Model Specification:

- **Formula**: The model formula, points ~ price + Rich + Soft + age + rich + year + soft + light + fruity + attractive + great + richness + dense + powerful + long + intense + (1 + price | variety), captures both fixed effects for individual predictors and random effects for variability in intercepts and price sensitivity across wine varieties. This hierarchical approach allows us to account for the specific influence that different wine varieties have on ratings and their distinct responses to pricing.

- **Priors:**
  - Regression Coefficients: Normal priors (normal(0, 1)) for regression coefficients were chosen to maintain neutrality in expectation, allowing the data itself to primarily inform the impact of predictors.

  - Intercept: An informative normal prior (normal(88, 5)) was set for the intercept to stabilize estimations around the central tendency observed in the preliminary data analysis.

  - Standard Deviations: Cauchy priors (cauchy(0, 2)) for the standard deviations of the random effects were selected to handle the potential wide variations among wine varieties, which is critical given their differing characteristics and market behaviours.

- **Family:**
  - Gaussian Distribution: I selected the Gaussian family for the response variable due to the continuous nature of wine ratings. This choice is supported by the distribution's properties, which suit the central limit theorem assumptions applicable to large sample sizes like ours.

- **Model Building Process:**

- Convergence and Sampling: I ensured model convergence with R-hat values close to 1.00, indicating adequate parameter space exploration. The No-U-Turn Sampler (NUTS), a refined form of Hamiltonian Monte Carlo, was utilized for its efficiency in navigating the high-dimensional space of our model parameters.

- **Benefits of Model Choices:**
  - Accuracy and Insight: The hierarchical model structure with appropriately chosen priors enhances both the accuracy and depth of insights from the analysis. It allows for detailed variability assessments within and across wine varieties, critical for developing nuanced understanding and targeted strategies.

  - Predictive Robustness: These configurations ensure that our model is not only robust in its predictive capabilities but also reflective of real-world complexities in wine rating dynamics. The use of sophisticated sampling methods like NUTS contributes to the reliability and credibility of our findings.

Results

Model overview

- **Model Hyperparameter Insights:**
  - **Variability Across Varieties**: Random effects show significant differences in base ratings (sd(Intercept) = 11.56) and price sensitivity (sd(price) = 6.58) across 35 wine varieties, emphasizing the unique market positioning of each variety.

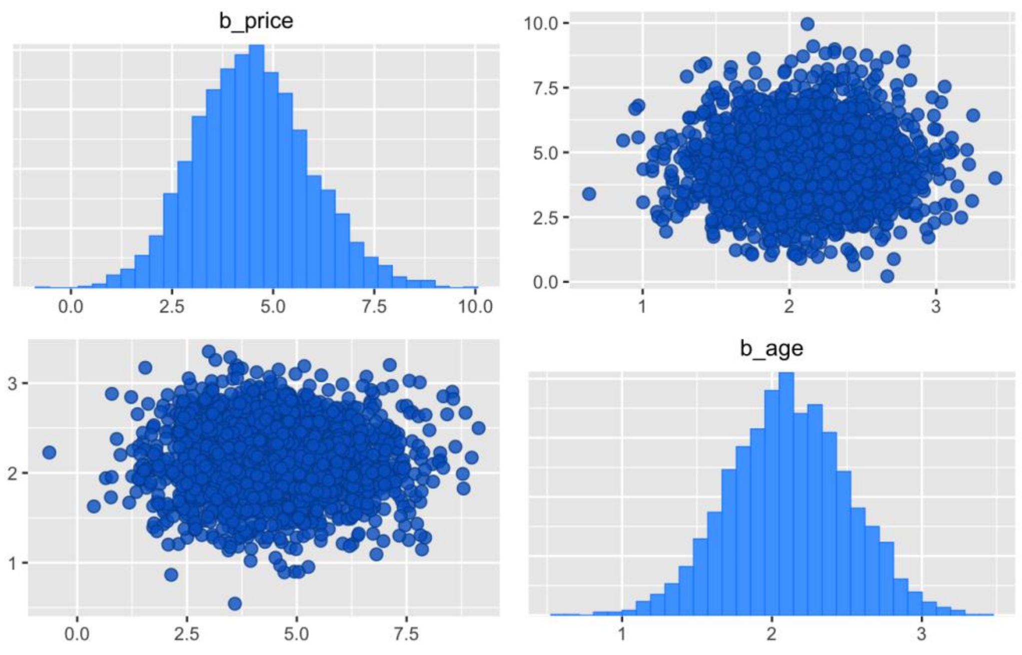
  - **Intercept and Price Sensitivity**: The substantial standard deviations indicate diverse responses to price changes, suggesting tailored pricing strategies for different varieties. The strong negative correlation (cor(Intercept,price) = -1.00) implies that higher-rated varieties are less influenced by price fluctuations.

- **Model Diagnostics:**
  - **Effective Sample Size (Bulk ESS and Tail ESS)**: Indicates adequate sample utilization for stable parameter estimates, despite some values being modest.

  - **Rhat (1.01)**: Close to 1, confirming good convergence and reliable estimates.

- **Implications of Hyperparameters**
  These findings are critical for:
  - **Strategic Decision Making**: Insights into variety-specific ratings and price sensitivities can guide targeted marketing and pricing strategies.

  - **Consumer Insights**: Helps in understanding consumer preferences and optimizing product offerings accordingly.



Key Insights

- **Effect Sizes of Predictors on Wine Ratings (Excluding Intercept)**:
  Our analysis revealed several significant predictors influencing wine ratings, as detailed in the visual representation of effect sizes:

  - **Age** has the most substantial positive effect, indicating that older wines tend to receive higher ratings, aligning with traditional views on wine aging and quality.

  - **Price** also shows a positive correlation, suggesting that higher-priced wines are generally perceived as better quality.

  - Descriptors from the wine reviews such as **"rich," "powerful," and "great"** are associated with higher ratings, reinforcing the importance of these attributes in how wines are perceived by experts. Similarly descriptors such as **“light” and “attractive”** are associated with lower rating.

The graph of effect sizes, excluding the intercept, provides clear visual evidence of how various predictors influence wine ratings. Each predictor's impact is shown with a confidence interval, highlighting the precision of our estimates.

- **Intercept and Price Effects Across Wine Varieties**:
  The scatter plot of intercept and price effects across different wine varieties provides insights into the inherent rating biases towards certain wine types and how sensitive these ratings are to price changes:

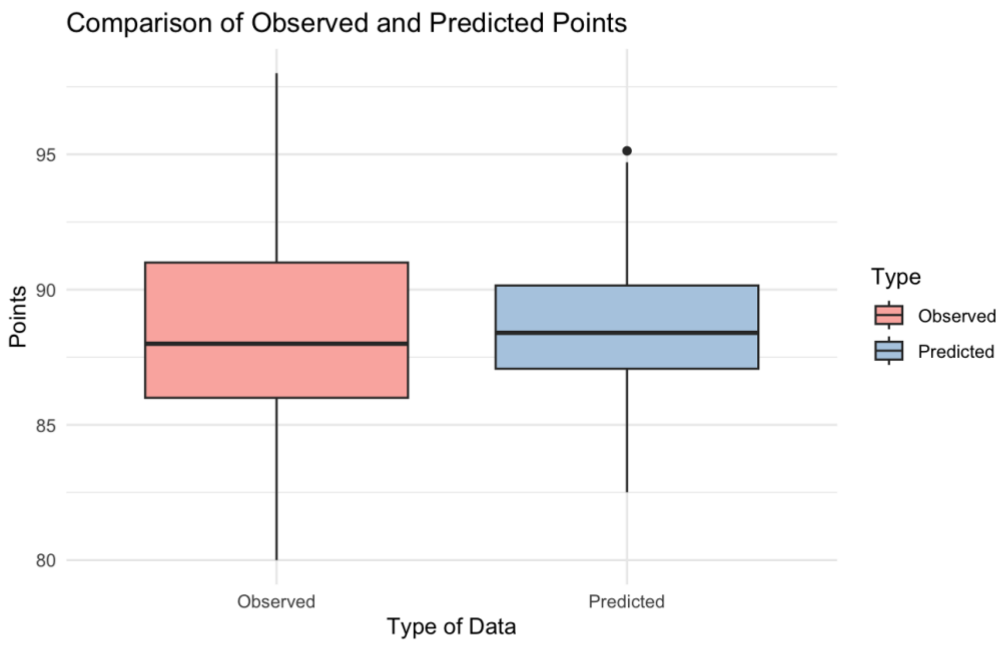
  - Varieties such as **Pinot Gris and Gewürztraminer** show high price sensitivity, indicating that their ratings significantly improve with higher prices.

  - Conversely, **Grenache and Sylvaner** exhibit low intercepts and less price sensitivity, suggesting that these varieties are generally rated lower regardless of price changes.

This analysis helps uncover hidden biases in wine ratings and can guide producers and marketers in adjusting their strategies based on the variety-specific price sensitivities.

Conclusion

- **Summary of Key Insights**:
  - Age and price are crucial determinants of wine quality perception.
  - Specific descriptive terms in wine reviews strongly influence expert ratings.
  - There is significant variability in how different wine varieties respond to price changes, indicating a need for variety-specific marketing and pricing strategies.



- **Key Results:**
    - Predictive accuracy as measured by RMSE (Root Mean Squared Error) was 1.913, indicating that the model predictions were reasonably close to the actual ratings.
    - The coefficient of determination (R^2) was 0.630, suggesting that approximately 63% of the variability in wine ratings was accounted for by the model.
    - By classifying the predicted rating into superior or not and comparing the results I got an accuracy of 79%.

- **Overall Evaluation**
    - The Bayesian approach was successful in capturing the complex interactions between various predictors and wine ratings, providing a robust framework for uncertainty estimation and model diagnostics:
    - The model diagnostics indicated good convergence and effective sample size across most parameters, as evidenced by Rhat values close to 1.
    - Posterior predictive checks confirmed that the model's predictions were consistent with the observed data distributions.

- **Recommendations for Further Research:**

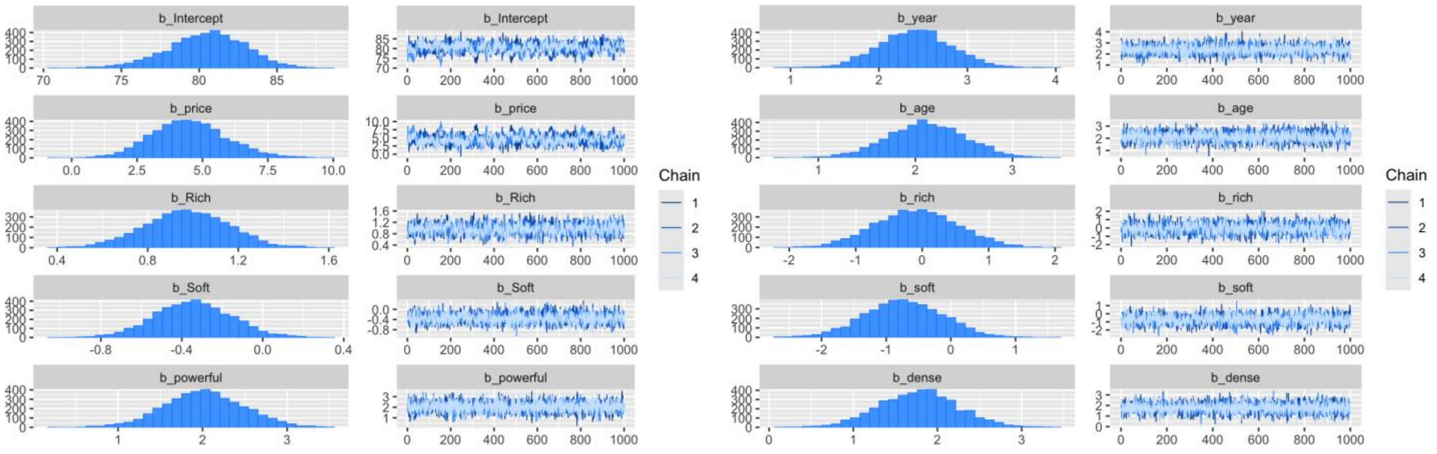
  While our model provided valuable insights, several enhancements could be explored to improve future studies:
    - Incorporating Additional Data: Including more diverse wine review data could help refine the predictions, especially for underrepresented varieties.
    - Model Complexity: Exploring more complex Bayesian models or alternative machine learning techniques like ensemble methods could potentially increase predictive accuracy.

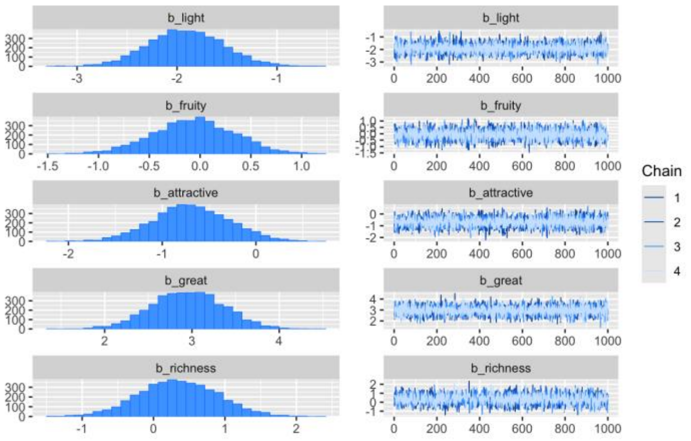
In conclusion, the Bayesian regression approach proved effective for this application, but as with any analytical model, there is room for further refinement and exploration to enhance predictive capabilities and insights into wine rating determinants. Future studies should consider these recommendations to build upon the foundation laid by this analysis.

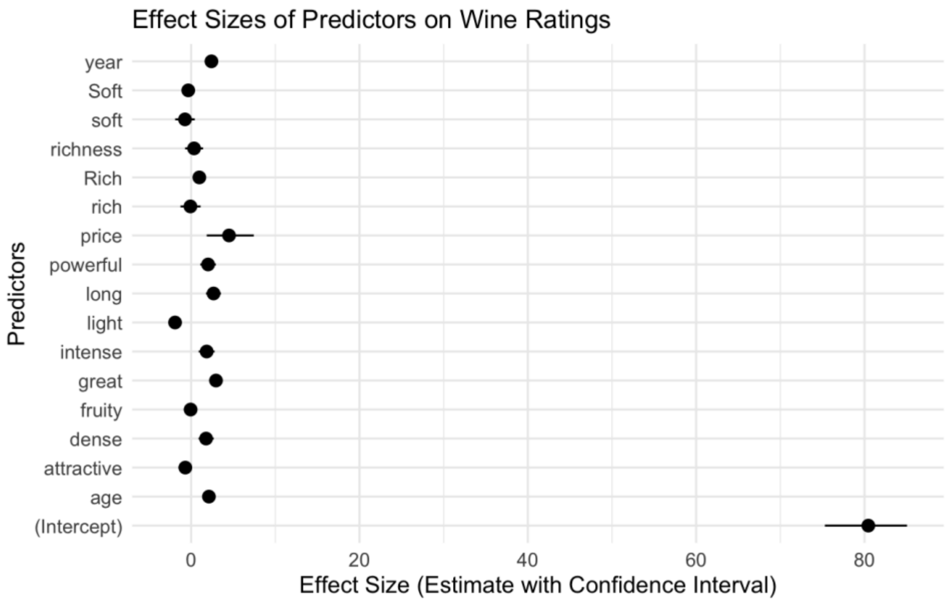
Appendix-

plots –

Posterior Distribution Histograms (left column) & Trace Plots (right column)







Code –

Pre-processing (python)-

```
import pandas as pd
import re
import nltk
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction import text
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')


# Step 1: Data Preprocessing
# Assuming 'data' is your DataFrame and 'description' is the column you want to analyze
# Make sure 'data' contains a column named 'description'
data = pd.read_csv('wine_review.csv')

# Define custom stop words
custom_stop_words = text.ENGLISH_STOP_WORDS.union(['year', 'wine', 'drink'])

# Initialize lemmatizer
lemmatizer = WordNetLemmatizer()

def get_wordnet_pos(treebank_tag):
    """
    Convert Treebank POS tags to WordNet POS tags
    """
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN  # Default to noun if not recognized

def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation and non-alphabetic characters
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    # Tokenize text
    tokens = nltk.word_tokenize(text)
    # Remove stop words and custom words, and perform lemmatization
    lemmatized_tokens = [lemmatizer.lemmatize(word, get_wordnet_pos(pos_tag))
                for word, pos_tag in nltk.pos_tag(tokens)
                if word not in custom_stop_words]
    # Join tokens back to text
    text = ' '.join(lemmatized_tokens)
    return text

# Apply preprocessing to the 'description' column
data['description'] = data['description'].apply(preprocess_text)

# Step 2: TF-IDF Vectorization
```

```
tfidf_vectorizer = TfidfVectorizer(max_features=100)  # You can adjust max_features as needed
tfidf_matrix = tfidf_vectorizer.fit_transform(data['description'])

# Step 3: Extract Top Features
feature_names = tfidf_vectorizer.get_feature_names_out()
feature_scores = tfidf_matrix.sum(axis=0).A1

# Combine feature names and scores into a DataFrame for better visualization
feature_df = pd.DataFrame({'Feature': feature_names, 'Score': feature_scores})
top_features = feature_df.sort_values(by='Score', ascending=False).head(100)

# Add top features as new columns to the original DataFrame
for feature in top_features['Feature']:
    data[feature] = tfidf_matrix[:, tfidf_vectorizer.vocabulary_[feature]].toarray()

# Remove the 'description' column
data.drop(columns=['description'], inplace=True)

# Print or further process the modified DataFrame
print(data.head())

import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
# Drop the 'points' column
# data.drop(columns=['superior_rating'], inplace=True)

# Transform the 'price' column to logarithmic scale
# Check skewness of the 'price' column before transformation
before_skewness = data['price'].skew()
print("Skewness before transformation:", before_skewness)
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
sns.histplot(data['price'], kde=True, color="blue")
plt.title('Original Price Distribution')

min_price = data['price'].min()
shift = 0

# If the minimum price is less than or equal to 0, shift all values by (1 - min_price)
if min_price <= 0:
    shift = 1 - min_price
    data['price'] += shift# Apply the Box-Cox transformation
price_transformed, best_lambda = stats.boxcox(data['price'])

# Store the transformed prices back into the DataFrame
data['price'] = price_transformed

# Print the lambda that was found to be best

# Check skewness of the 'price' column after transformation
after_skewness = data['price'].skew()
print("Skewness after transformation:", after_skewness)
plt.figure(figsize=(14, 6))
plt.subplot(1, 2, 1)
```

```
sns.histplot(data['price'], kde=True, color="blue")
plt.title('Transformed Price Distribution')
# Print or further process the modified DataFrame



from wordcloud import WordCloud

print(data.head())
numeric_columns = data.select_dtypes(include=['number'])

# Calculate correlation coefficients with 'superior_rating' for numeric columns
correlation = numeric_columns.corrwith(data['points']).abs()

# Select features with correlation greater than magnitude of 0.15
selected_features = correlation[correlation > 0.15].index.tolist()
selected_features.append('variety')


# Print selected features
print("Selected features:")
print(selected_features)
final_df=data[selected_features]
final_df.to_csv('final_df.csv', index=False)
```

Modelling and analysis (R):

```
# Loading necessary libraries
library(readr)       # For reading CSV files
library(tm)          # Text mining capabilities
library(SnowballC)   # Provides word stemming
library(wordcloud)   # Visualization of text data
library(RColorBrewer) # Enhances color selection for plots
library(dplyr)       # Data manipulation
library(data.table)  # Efficient data handling
library(textstem)    # Provides lemmatization
library(caret)       # Machine learning methods
library(tidyverse)   # Data manipulation and visualization
library(brms)        # Bayesian regression modeling

# Load the dataset
wine_reviews <- read_csv("final_df.csv")
names(wine_reviews)
wine_reviews$variety <- as.factor(wine_reviews$variety)

correlation <- cor(wine_reviews[sapply(wine_reviews, is.numeric)], use = "pairwise.complete.obs")
correlation <- abs(correlation['points', ])

# Remove 'points' and 'superior_rating' from the correlation vector
correlation <- correlation[!names(correlation) %in% c('points', 'superior_rating')]
# Generate the word cloud
wordcloud(names(correlation), correlation, scale=c(3,0.5), max.words=100, random.order=FALSE, rot.per=0.35,
colors=brewer.pal(8, "Dark2"))


library(caret)
set.seed(123)
```

```
# Partition the data into training and testing sets (80% training, 20% testing)
split <- createDataPartition(wine_reviews$points, p = 0.80, list = FALSE)
train_data <- wine_reviews[split, ]
test_data <- wine_reviews[-split, ]

# Store the 'superior_rating' from test data for later use in validation
superior_rating <- test_data$superior_rating

# Remove the 'superior_rating' column from both training and testing datasets
# This ensures it does not influence the regression model
train_data$superior_rating <- NULL
test_data$superior_rating <- NULL

brms_model <- brm(
 formula = points ~ price + Rich + Soft + age + rich + year + soft + light + fruity + attractive + great+ richness+ dense +
powerful + long  +intense + (1 + price | variety),
 data = train_data,
 family = gaussian(),
 prior = c(
  set_prior("normal(88, 5)", class = "Intercept"),
  set_prior("normal(0, 1)", class = "b"),
  set_prior("cauchy(0, 2)", class = "sd"),
  set_prior("cauchy(0, 2)", class = "sigma")
 ),
 chains = 4,
 iter = 2000,
  # warmup = 1000,
 control = list(adapt_delta = 0.95)
)


summary(brms_model)

library(bayesplot)
color_scheme_set("brightblue")
mcmc_trace(brms_model, pars = c("b_Intercept", "b_price"))
mcmc_pairs(brms_model, pars = c("b_price", "b_age"))

pp_check(brms_model)

predicted_points_test <- fitted(brms_model, newdata = test_data, summary = FALSE)
predicted_points_test <- colMeans(predicted_points_test)  # Assuming one observation per row
rmse_value_test <- sqrt(mean((predicted_points_test - test_data$points)^2))
ss_res_test <- sum((predicted_points_test - test_data$points)^2)
ss_tot_test <- sum((test_data$points - mean(test_data$points))^2)
r_squared_test <- 1 - ss_res_test / ss_tot_test
cat("Testing RMSE:", rmse_value_test, "\n")
cat("Testing R^2:", r_squared_test, "\n")

fixef(brms_model)
ranef(brms_model)  # To view random effects deviations by variety

# conditional_effects(brms_model)
plot(brms_model, variable =c("b_Intercept", "b_price", "b_Rich", "b_Soft","b_powerful"))
plot(brms_model, variable =c("b_year", "b_age", "b_rich", "b_soft","b_dense"))
plot(brms_model, variable =c("b_light", "b_fruity", "b_attractive", "b_great","b_richness"))
```

```
predicted_classifications <- ifelse(predicted_points_test >= 90, 1, 0)
conf_matrix <- confusionMatrix(as.factor(predicted_classifications), as.factor(superior_rating))
cat("Accuracy using caret:", conf_matrix$overall['Accuracy'], "\n")
cat("Sensitivity (Recall):", conf_matrix$byClass['Sensitivity'], "\n")
cat("Specificity:", conf_matrix$byClass['Specificity'], "\n")
cat("Precision:", conf_matrix$byClass['Precision'], "\n")
cat("F1 Score:", conf_matrix$byClass['F1'], "\n")

random_effects <- ranef(brms_model)
random_effects <- random_effects$variety

# If the above doesn't work, use the following to try and see the content
print(str(random_effects))
intercept_df <- data.frame(
  Variety = dimnames(random_effects)[[1]],  # Extract varieties names
  Intercept_Estimate = random_effects[, "Estimate", "Intercept"],
  Intercept_EstError = random_effects[, "Est.Error", "Intercept"],
  Intercept_Q2.5 = random_effects[, "Q2.5", "Intercept"],
  Intercept_Q97.5 = random_effects[, "Q97.5", "Intercept"]
)

# Create a dataframe for price estimates
price_df <- data.frame(
  Variety = dimnames(random_effects)[[1]],  # Extract varieties names
  Price_Estimate = random_effects[, "Estimate", "price"],
  Price_EstError = random_effects[, "Est.Error", "price"],
  Price_Q2.5 = random_effects[, "Q2.5", "price"],
  Price_Q97.5 = random_effects[, "Q97.5", "price"]
)

# Merge dataframes by 'Variety'
combined_df <- merge(intercept_df, price_df, by = "Variety")

combined_plot <- ggplot(combined_df, aes(x = Intercept_Estimate, y = Price_Estimate, label = Variety)) +
  geom_point() +
  geom_text(aes(label=Variety), hjust=1.1, vjust=1.1, angle=45, size=3) +
  labs(title = "Intercept and Price Effects Across Wine Varieties",
       x = "Intercept Estimate",
       y = "Price Estimate") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
        axis.text.y = element_text(size = 8)) +
  expand_limits(x = c(min(combined_df$Intercept_Estimate) - 2, max(combined_df$Intercept_Estimate) + 2),
                y = c(min(combined_df$Price_Estimate) - 2, max(combined_df$Price_Estimate) + 2))

print(combined_plot)

# Ensure you have the necessary package
if (!requireNamespace("broom.mixed", quietly = TRUE)) {
  install.packages("broom.mixed")
}
library(broom.mixed)

# Extracting coefficients and confidence intervals
model_coef <- tidy(brms_model, effects = "fixed", conf.int = TRUE)
```

```r
# Check the structure
print(model_coef)

coef_plot <- ggplot(model_coef, aes(x = term, y = estimate, ymin = conf.low, ymax = conf.high)) +
 geom_pointrange() +
 coord_flip() +  # Flip coordinates for easier reading
 labs(title = "Effect Sizes of Predictors on Wine Ratings",
     x = "Predictors",
     y = "Effect Size (Estimate with Confidence Interval)") +
 theme_minimal() +
 theme(axis.text.x = element_text(size = 10),
      axis.title.x = element_text(size = 12),
      axis.text.y = element_text(size = 10),
      axis.title.y = element_text(size = 12))

# Print the plot
print(coef_plot)



predictor_coef <- model_coef %>%
 filter(term != "(Intercept)")  # Adjust this line if the intercept has a different label


coef_plot <- ggplot(predictor_coef, aes(x = term, y = estimate, ymin = conf.low, ymax = conf.high)) +
 geom_pointrange() +
 coord_flip() +  # Flip coordinates for easier reading
 labs(title = "Effect Sizes of Predictors on Wine Ratings (Excluding Intercept)",
     x = "Predictors",
     y = "Effect Size (Estimate with Confidence Interval)") +
 theme_minimal() +
 theme(axis.text.x = element_text(size = 10),
      axis.title.x = element_text(size = 12),
      axis.text.y = element_text(size = 10),
      axis.title.y = element_text(size = 12))

# Print the plot
print(coef_plot)


points_data <- data.frame(
 Score = c(test_data$points, predicted_points_test),
 Type = factor(rep(c("Observed", "Predicted"), each = length(test_data$points)))
)

# Using ggplot2 to create a box plot

# Create the box plot comparing observed and predicted points
box_plot <- ggplot(points_data, aes(x = Type, y = Score, fill = Type)) +
 geom_boxplot() +
 labs(title = "Comparison of Observed and Predicted Points",
     x = "Type of Data",
     y = "Points") +
 scale_fill_brewer(palette = "Pastel1") +
 theme_minimal()

# Print the box plot
```

```
print(box_plot)

test_data <- test_data %>%
  mutate(predicted_rating = predicted_points_test)

# Now filter for specific varieties
specific_varieties <- test_data %>%
  filter(variety %in% c("Pinot Gris", "Bordeaux-style Red Blend"))

ggplot(specific_varieties, aes(x = predicted_rating, fill = variety)) +
  geom_density(alpha = 0.5) +
  labs(title = "Probability Distribution of Predicted Ratings for
       Pinot Gris and Bordeaux-style Red Blend",
       x = "Predicted Rating",
       y = "Density") +
  scale_fill_brewer(palette = "Set1") +
  theme_minimal() +
  theme(legend.title = element_blank())


test_data <- test_data %>%
  mutate(predicted_rating = predicted_points_test,
         rich_present = ifelse(rich > 0, "Rich Present", "Rich Absent"),
         light_present = ifelse(light > 0, "Light Present", "Light Absent"))

# You can choose to focus on one descriptor at a time or compare them; here's how to plot for 'rich'
ggplot(test_data, aes(x = predicted_rating, fill = rich_present)) +
  geom_density(alpha = 0.5) +
  labs(title = "Probability Distribution of Predicted Ratings for 'Rich' Descriptor",
       x = "Predicted Rating", y = "Density") +
  scale_fill_brewer(palette = "Set1") +
  theme_minimal() +
  theme(legend.title = element_blank())


ggplot(test_data, aes(x = predicted_rating, fill = light_present)) +
  geom_density(alpha = 0.5) +
  labs(title = "Probability Distribution of Predicted Ratings for 'Light' Descriptor",
       x = "Predicted Rating", y = "Density") +
  scale_fill_brewer(palette = "Set1") +
  theme_minimal() +
  theme(legend.title = element_blank())
```