**Assignment: Predicting future outcomes**

We are a team of data analysts contracted by Turtle Games, a game manufacturer and retailer with a global customer base. The company manufactures and sells its own products, along with sourcing and selling products manufactured by other companies. Its product range includes books, board games, video games, and toys. The company collects data from sales as well as customer reviews. Turtle Games has a business objective of improving overall sales performance by utilising customer trends.

I used Python and R for my analysis. These are both open-source programming languages which have a large and active community of users and developers. They have extensive libraries and packages for data analysis, machine learning, statistics and visualisation such as pandas, NumPy, scikit-learn and Matplotlib in Python, while R has packages like dplyr, ggplot2 and caret. R and Python both provide interactive environments. Jupyter notebooks for Python and RMarkdown for R allow you to combine code, visualizations, and narrative in a single document, making it easier to share and communicate your findings.

**Analytical Approach - Python:**

In Python, I followed a systematic approach to import, clean, and analyse the data for Turtle Games. The primary objective was to gain insights into sales, product performance, and customer behaviour. Here's a detailed description of the processes:

I began by importing the necessary libraries, including pandas, nltk, wordcloud, and matplotlib, to facilitate data manipulation, text analysis, and visualisation. I loaded the data from a CSV file into a pandas DataFrame. Data cleaning started by checking for missing values, duplicates, and outliers. Fortunately, the dataset was relatively clean, requiring minimal cleaning. This was done using the code below;

```python
import pandas as pd

# Load the turtle_reviews.csv file, explore the data and create a new DataFrame (e.g. reviews).
# Specify the path to your CSV file
csv_file_path = "turtle_reviews.csv"

# Load the CSV file into a DataFrame
reviews = pd.read_csv(csv_file_path)

# Explore the data
# Display the first few rows
print(reviews.head())

# Get basic statistics of the numerical columns
print(reviews.describe())

# Check the data types of each column
print(reviews.dtypes)

# Check for missing values
print(reviews.isnull().sum())
```

To remove redundant columns from the DataFrame I used the drop method reviews = reviews.drop(["language", "platform"], axis=1). To rename columns to more user-friendly names, I used the rename method using the code;

```python
reviews.rename(columns={"language": "remuneration", "platform": "spending_score"}, inplace=True)
```

I saved the reviews DataFrame to a new CSV file called "clean_turtle_reviews.csv."

```python
clean_csv_path = "clean_turtle_reviews.csv"

# Save the clean DataFrame to a CSV file
reviews.to_csv(clean_csv_path, index=False)
```

I proceeded to evaluate the possible linear relationships between loyalty points, age, remuneration, and spending scores, and to determine whether these variables can be used to predict loyalty points. I created the OLS (Ordinary Least Squares) multiple linear regression with the statsmodels library in Python using the following code;

```python
# To evaluate the possible linear relationships between loyalty points and age, remuneration,
# and spending scores, and to determine whether these variables can be used to predict loyalty points,
# you can use multiple linear regression with the statsmodels library in Python
import statsmodels.api as sm

# Define the independent variables (predictors)
X = imported_reviews[["age", "remuneration (k£)", "spending_score (1-100)"]]

# Add a constant to the model (intercept)
X = sm.add_constant(X)

# Define the dependent variable (target)
y = imported_reviews["loyalty_points"]

# Fit the multiple linear regression model
model = sm.OLS(y, X).fit()

# Get the summary of the regression model
print(model.summary())
```

I used Seaborn to create a scatterplot showing the relationship between "remuneration" and "spending_score.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a scatterplot to visualize the relationship
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df2, x="remuneration (k£)", y="spending_score (1-100)")
plt.title("Scatterplot of Remuneration vs. Spending Score")
plt.xlabel("Remuneration (k£)")
plt.ylabel("Spending Score (1-100)")
plt.grid(True)
plt.show()

# Create a pair plot to visualize pairwise relationships
sns.pairplot(df2)
plt.suptitle("Pair Plot of Remuneration vs. Spending Score", y=1.02)
plt.show()
```

I determined the optimal number of clusters for k-means clustering, using the Silhouette method and the Elbow method, using the sklearn.cluster, sklearn.metrics. I specified a range of potential cluster numbers to explore, and calculated the Silhouette scores and inertias for each cluster number by fitting k-means clustering models to the data.

In the Silhouette subplot, we plot the Silhouette scores for different cluster numbers. The optimal number of clusters is typically associated with the highest Silhouette score. In the Elbow subplot, we plot the inertias for different cluster numbers. The "elbow" point, where the inertia begins to level off, indicates a good choice for the number of clusters.

```
# Based on insights from the Elbow and Silhouette methods, consider at least three values for k.
k_values = [3, 4, 5]
# We can adjust these values based on the insights.

# Initialize a list to store cluster assignments for each k
cluster_assignments = []

# Plot the predicted k-means for each value of k
plt.figure(figsize=(12, 4))

for i, k in enumerate(k_values, 1):
    # Fit a k-means model with the current value of k
    kmeans = KMeans(n_clusters=k, random_state=0)
    cluster_labels = kmeans.fit_predict(df2)
    cluster_assignments.append(cluster_labels)

    # Create a subplot for each k
    plt.subplot(1, len(k_values), i)

    # Plot the predicted clusters
    plt.scatter(df2["remuneration (k£)"], df2["spending_score (1-100)"], c=cluster_labels, cmap='viridis')
    plt.title(f"K-Means Clustering (k={k})")
    plt.xlabel("Remuneration (k£)")
    plt.ylabel("Spending Score (1-100)")
```

```
plt.tight_layout()
plt.show()
```

To prepare the data for Natural Language Processing (NLP) I changed the data to lower case using the .str.lower() method, and joined the elements in the review and summary columns. I defined a remove_punctuation function to remove punctuation from text using str.translate with string.punctuation. I dropped duplicate rows in both columns using .drop_duplicates().

```
# 1. Change to Lower Case
df2["review"] = df2["review"].str.lower()
df2["summary"] = df2["summary"].str.lower()

# 2. Join Elements in Each Column (Optional)
# To combine "review" and "summary" into a single text column:
df2["combined_text"] = df2["review"] + " " + df2["summary"]

# 3. Replace Punctuation (Removing Punctuation)
import string

def remove_punctuation(text):
    translator = str.maketrans('', '', string.punctuation)
    return text.translate(translator)

df2["review"] = df2["review"].apply(remove_punctuation)
df2["summary"] = df2["summary"].apply(remove_punctuation)

# 4. Drop Duplicates
df2 = df2.drop_duplicates(subset=["review", "summary"])

# Print a sample of the cleaned data
print("Cleaned DataFrame:")
print(df2.head())
```

I conducted text analysis on customer reviews and summaries using the NLTK library. Tokenization was applied to split text into individual words for frequency distribution and word cloud generation. Stopwords were removed to focus on meaningful words. I created word clouds to visually represent common words and their sentiment in reviews. The libraries 'wordcloud' and 'stopwords' must be imported to ensure these functions work.

```
import pandas as pd
import nltk
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Using the DataFrame 'df2' with 'review' and 'summary' columns
# Create a copy of the DataFrame
df_copy = df2.copy()

# Tokenization using nltk
nltk.download("punkt")
# Download the tokenizer data if not already downloaded

# Tokenize the "review" and "summary" columns
df_copy["review_tokens"] = df_copy["review"].apply(nltk.word_tokenize)
df_copy["summary_tokens"] = df_copy["summary"].apply(nltk.word_tokenize)

# Create and plot word clouds for the "review" column
review_wordcloud = WordCloud(width=800, height=400, background_color="white").generate(" ".join(df_copy["review"]))
plt.figure(figsize=(10, 5))
plt.imshow(review_wordcloud, interpolation="bilinear")
plt.title("Word Cloud for Review Column")
plt.axis("off")
plt.show()
```

```
# Create and plot word clouds for the "summary" column
summary_wordcloud = WordCloud(width=800, height=400, background_color="white").generate(" ".join(df_copy["summary"]))
plt.figure(figsize=(10, 5))
plt.imshow(summary_wordcloud, interpolation="bilinear")
plt.title("Word Cloud for Summary Column")
plt.axis("off")
plt.show()
```

I applied sentiment analysis and printed the desired top 20 positive and negative reviews and summaries based on sentiment scores. I created a SentimentIntensityAnalyzer from the nltk.sentiment module to calculate sentiment scores for each review in the DataFrame, added a new column 'sentiment_score' to the DataFrame to store the sentiment scores for each review, sorted the DataFrame based on the 'sentiment_score' column to get the most positive and most negative reviews.

```python
# We use the pandas library to read the CSV file and then apply the sentiment analysis and sorting logic.

import pandas as pd
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

# Load the CSV file into a DataFrame
df = pd.read_csv('turtle_reviews.csv')

# Create a SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()

# Calculate sentiment scores for each review
df['sentiment_score'] = df['review'].apply(lambda x: sia.polarity_scores(x)['compound'])

# Sort the reviews based on sentiment scores (from most positive to most negative)
sorted_reviews = df.sort_values(by='sentiment_score', ascending=False)
```

```python
# Print the top 20 positive reviews and summaries
print("Top 20 Positive Reviews and Summaries:")
for i, row in enumerate(sorted_reviews.head(20).itertuples(), 1):
    print(f"{i}. Summary: {row.summary}")
    print(f"   Review: {row.review}")
    print(f"   Sentiment Score: {row.sentiment_score}\n")

# Print the top 20 negative reviews and summaries
print("Top 20 Negative Reviews and Summaries:")
for i, row in enumerate(sorted_reviews.tail(20).itertuples(), 1):
    print(f"{i}. Summary: {row.summary}")
    print(f"   Review: {row.review}")
    print(f"   Sentiment Score: {row.sentiment_score}\n")
```

I used Matplotlib to create visualizations, including word clouds, histograms, boxplots, and scatterplots. Visualizations played a crucial role in presenting insights effectively. An example of the code used to plot a scatter plot;

```python
# To plot the linear regression and add a regression line to your data, we use the matplotlib library in Python
import matplotlib.pyplot as plt
import numpy as np

# Extract the coefficients (including the constant) from the model
coefficients = model.params

# Extract the independent variables
age = imported_reviews["age"]
remuneration = imported_reviews["remuneration (k£)"]
spending_score = imported_reviews["spending_score (1-100)"]

# Create a scatter plot of the data points
plt.scatter(age, imported_reviews["loyalty_points"], label="Age vs Loyalty Points", alpha=0.5)
plt.scatter(remuneration, imported_reviews["loyalty_points"], label="Remuneration vs Loyalty Points", alpha=0.5)
plt.scatter(spending_score, imported_reviews["loyalty_points"], label="Spending Score vs Loyalty Points", alpha=0.5)

# Generate a regression line for each independent variable
reg_line_age = coefficients["const"] + coefficients["age"] * age
reg_line_remuneration = coefficients["const"] + coefficients["remuneration (k£)"] * remuneration
reg_line_spending_score = coefficients["const"] + coefficients["spending_score (1-100)"] * spending_score

# Plot the regression lines
plt.plot(age, reg_line_age, color="red", label="Age Regression Line")
plt.plot(remuneration, reg_line_remuneration, color="green", label="Remuneration Regression Line")
plt.plot(spending_score, reg_line_spending_score, color="blue", label="Spending Score Regression Line")
```

```python
# Set labels and legend
plt.xlabel("Independent Variables")
plt.ylabel("Loyalty Points")
plt.legend(loc="upper right")

# Show the plot
plt.show()
```

**Analytical Approach - R:**

Following a similar structured approach in R, I set the working directory to access the CSV file containing sales data. I imported the CSV file into R using

the read.csv() function. Data cleaning involved checking for missing values and duplicates, similar to the Python process.

I performed Ordinary Least Squares (OLS) regression analysis to model the relationship between loyalty_points and predictor variables (age, remuneration, spending_score). Model diagnostics and statistical tests, such as R-squared and p-values, were used to evaluate model fit and significance.

EDA was conducted using summary statistics, histograms, and visualisations to understand the distribution and relationships in the loyalty_points dataset.

Predictions were made for various scenarios to assess the impact of predictor variables on loyalty_points.

I examined correlations between sales columns (North American, European, and global sales) to understand relationships and dependencies.

To determine the normality of the sales data, I installed the moments package, created Q-Q plots, performed the Shapiro-Wilk test, calculated skewness and kurtosis, and checked for correlations between sales data columns, using R's statistical functions and visualisation libraries eg ggplot2 and stats. The Q-Q plots visually show how closely the data follows a normal distribution. Positive skewness indicates a right-skewed distribution, while negative skewness indicates a left-skewed distribution. Kurtosis measures the "tailedness" of the distribution. The correlation matrix shows the correlation coefficients between the sales data columns. Positive values indicate positive correlation, negative values indicate negative correlation, and values close to zero indicate weak or no correlation. Used the code below to calculate skewness and kurtosis;

```
install.packages("moments")
library(moments)

# Calculate skewness and kurtosis for all sales data columns
skew_na_sales <- skewness(turtle_sales$NA_Sales)
kurt_na_sales <- kurtosis(turtle_sales$NA_Sales)

skew_eu_sales <- skewness(turtle_sales$EU_Sales)
kurt_eu_sales <- kurtosis(turtle_sales$EU_Sales)

skew_global_sales <- skewness(turtle_sales$Global_Sales)
kurt_global_sales <- kurtosis(turtle_sales$Global_Sales)

# Print skewness and kurtosis values
cat("NA Sales Skewness:", skew_na_sales, "Kurtosis:", kurt_na_sales, "\n")
cat("EU Sales Skewness:", skew_eu_sales, "Kurtosis:", kurt_eu_sales, "\n")
cat("Global Sales Skewness:", skew_global_sales, "Kurtosis:", kurt_global_sales, "\n")
```

To create a multiple linear regression model to predict "Global_Sales" based on "NA_Sales" and "EU_Sales", I defined the sums of NA_Sales and EU_Sales for each scenario and then used the predict() function with the multiple linear regression model (multiple_linear_model) to predict Global_Sales.

```r
# Select only numeric columns
numeric_columns <- turtle_sales[, sapply(turtle_sales, is.numeric)]

# Calculate correlations between numeric columns
correlation_matrix <- cor(numeric_columns)

# Print the correlation matrix
correlation_matrix

# Create a multiple linear regression model
multiple_linear_model <- lm(Global_Sales ~ NA_Sales + EU_Sales, data = turtle_sales)

# View the summary of the multiple linear regression model
summary(multiple_linear_model)
```

To predict global sales based on provided values of NA_Sales and EU_Sales sums and compare the predictions to the observed values, I used the multiple linear regression model.

```r
# Load necessary libraries
library(dplyr)

# Define the sums of NA_Sales and EU_Sales for each scenario
scenarios <- data.frame(NA_Sales = c(34.02, 3.93, 2.73, 2.26, 22.08),
                        EU_Sales = c(23.80, 1.56, 0.65, 0.97, 0.52))

# Predict Global_Sales for each scenario using the multiple linear regression model
predictions <- predict(multiple_linear_model, newdata = scenarios)

# Create a data frame to compare observed and predicted values
comparison <- data.frame(Observed_Global_Sales = c(67.80, 33.00, 29.40, 27.10, 25.70),
                         Predicted_Global_Sales = predictions)

# Print the comparison
print(comparison)
```

Both Python and R provided powerful tools for data analysis and visualization. The choice of libraries and functions in both languages was based on their suitability for specific tasks, ensuring effective data preparation and analysis aligned with Turtle Games' objectives.

**Visualizations and Insights:**

In the analysis for Turtle Games, I employed a variety of visualisations to provide clear insights into sales data, customer behaviour, and sentiment. Each visualisation was chosen with a specific rationale in mind, aligning with Turtle Games' business objectives:

Word Clouds for Customer Sentiment: Word clouds were used to visually represent the most common words in customer reviews and summaries. This allowed for a quick understanding of prevalent customer sentiments.

Positive words like "great," "excellent," and "amazing" were identified, indicating positive customer sentiment. Conversely, negative words like "terrible" and "bad" revealed areas for improvement.

Histograms were employed to illustrate the distribution of sales data (NA_Sales, EU_Sales, Global_Sales). This helped in identifying sales trends and outliers. The

histograms showed that most sales values were concentrated in lower ranges, with a few high-value outliers. This information is valuable for sales strategy and resource allocation.

Scatterplots were used to explore the relationship between individual products and sales performance, helping to identify top-performing products. Some products exhibited a strong positive correlation with sales, suggesting their significance in driving revenue. Others showed little to no correlation, indicating areas for potential improvement or product optimization.

A multiple linear regression plot was used to visualize the relationship between loyalty points and predictor variables (age, remuneration, spending score). The plot illustrated how each predictor variable influenced loyalty points. Higher remuneration and spending scores were associated with increased loyalty points, while age had a more moderate impact. This information can inform loyalty program strategies.

A correlation matrix was employed to visualize the relationships between North American, European, and global sales. The matrix revealed strong positive correlations between regional and global sales. This suggests that strong performance in one region often translates into higher global sales. Targeted marketing efforts in specific regions can yield global benefits.

The summary of the regression analysis provided valuable insights into the impact of predictor variables on loyalty points. The high R-squared value of 84% indicated that the model explained a significant portion of the variance in loyalty points. This suggests that age, remuneration, and spending score are strong predictors of customer loyalty.

Overall, the selected visualizations served the purpose of providing actionable insights for Turtle Games. They helped identify positive and negative customer sentiments, understand sales distribution, pinpoint product performance, and assess the impact of predictor variables on loyalty. These insights can inform marketing campaigns, product strategies, and customer engagement initiatives to improve overall sales and customer satisfaction.

**Patterns and Predictions:**

Through the analysis of Turtle Games' data, several patterns and predictions have emerged that are directly relevant to the business scenario and align with Turtle Games' objectives:

Sales Patterns:

Patterns in the sales data showed that most sales values were concentrated in lower ranges, with occasional high-value outliers. This suggests that a majority of products generate modest revenue, while a few high-performing products

significantly contribute to overall sales. Turtle Games can focus on optimizing and promoting these top-performing products.

Product Impact:

The analysis revealed distinct patterns in the impact of individual products on sales. Some products exhibited strong positive correlations with sales, indicating their importance in revenue generation. Identifying these products allows Turtle Games to allocate resources and marketing efforts more effectively.

Customer Sentiment:

Customer sentiment analysis highlighted patterns in customer reviews and summaries. Positive sentiments like "great" and "excellent" were common, indicating satisfaction with certain aspects of the products. However, negative sentiments like "terrible" pointed to areas for improvement. Turtle Games can use this feedback to enhance product quality.

Loyalty Points Prediction:

The multiple linear regression model predicted loyalty points based on customer age, remuneration, and spending score. Patterns in the model showed that remuneration and spending score had a significant positive impact on loyalty points, suggesting that incentivising spending and engagement can enhance customer loyalty.

Sales Correlation:

The correlation analysis demonstrated strong positive correlations between regional (NA and EU) and global sales. Patterns indicated that success in one region often translates to higher global sales. Turtle Games can strategically target marketing campaigns in specific regions to drive global revenue.

Future Predictions:

Using the regression model, Turtle Games can predict the loyalty points of customers based on their age, remuneration, and spending score. This prediction can inform personalized marketing strategies, loyalty program incentives, and customer retention efforts.

These patterns and predictions offer actionable insights for Turtle Games, helping them make informed decisions regarding product optimization, marketing campaigns, and customer engagement strategies. By leveraging these insights, Turtle Games can work toward improving overall sales performance and customer satisfaction, ultimately aligning with their business objectives.