Customer Sentiment Analysis and Feedback Insights

```
import pandas as pd


# Load the dataset
dataset_path = extract_to_dir + 'purchase behaviour dataset.csv'
df = pd.read_csv(dataset_path)


# Display the first few rows of the dataset to understand its structure
df.head()
```

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream
4	1005	MIDAGE SINGLES/COUPLES	Mainstream

```
# Analyzing the distribution of customers across different life stages and
premium categories


# Count of customers in each life stage
life_stage_counts = df['LIFESTAGE'].value_counts()


# Count of customers in each premium category
premium_customer_counts = df['PREMIUM_CUSTOMER'].value_counts()


life_stage_counts, premium_customer_counts
```

```
(RETIREES                 14805
 OLDER SINGLES/COUPLES    14609
 YOUNG SINGLES/COUPLES    14441
 OLDER FAMILIES            9780
 YOUNG FAMILIES            9178
 MIDAGE SINGLES/COUPLES    7275
```

```
NEW FAMILIES               2549
Name: LIFESTAGE, dtype: int64,
Mainstream    29245
Budget        24470
Premium       18922
Name: PREMIUM_CUSTOMER, dtype: int64)
```

```python
# Step 1: Create the Hypothetical Dataset

# Sample data
hypothetical_data = {
    'Customer_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Product_ID': [101, 102, 103, 104, 105, 106, 107, 108, 109, 110],
    'Review_Text': [
        "This product was amazing, I loved it!",
        "Not bad, but could be better.",
        "Worst purchase I've ever made.",
        "Decent quality, not what I expected but okay.",
        "Excellent product, will buy again!",
        "Broke after the first use, very disappointed.",
        "Average, not much to say.",
        "Exceeded my expectations, great quality!",
        "Terrible, would not recommend to anyone.",
        "Good product, but the shipping took too long."
    ],
    'Rating': [5, 3, 1, 3, 5, 1, 3, 5, 1, 4]
}

# Convert to DataFrame
hypothetical_reviews_df = pd.DataFrame(hypothetical_data)

# Assign sentiment labels based on ratings
hypothetical_reviews_df['Sentiment'] = np.select(
    [
```

```
        hypothetical_reviews_df['Rating'] >= 4,
        hypothetical_reviews_df['Rating'] == 3,
        hypothetical_reviews_df['Rating'] <= 2
    ],
    [
        'Positive',
        'Neutral',
        'Negative'
    ]
)

hypothetical_reviews_df
```

```
Customer_ID  Product_ID                                    Review_Text  \
0            1         101          This product was amazing, I loved it!
1            2         102                   Not bad, but could be better.
2            3         103                 Worst purchase I've ever made.
3            4         104  Decent quality, not what I expected but okay.
4            5         105             Excellent product, will buy again!
5            6         106  Broke after the first use, very disappointed.
6            7         107                       Average, not much to say.
7            8         108       Exceeded my expectations, great quality!
8            9         109       Terrible, would not recommend to anyone.
9           10         110  Good product, but the shipping took too long.

   Rating Sentiment
0       5  Positive
1       3   Neutral
2       1  Negative
3       3   Neutral
4       5  Positive
5       1  Negative
6       3   Neutral
7       5  Positive
```

```
8       1  Negative
9       4  Positive
```

```
# Step 2: Preprocess the Data and Vectorize the Text

# Initialize the TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english')

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    hypothetical_reviews_df['Review_Text'],
    hypothetical_reviews_df['Sentiment'],
    test_size=0.3,
    random_state=42
)
```

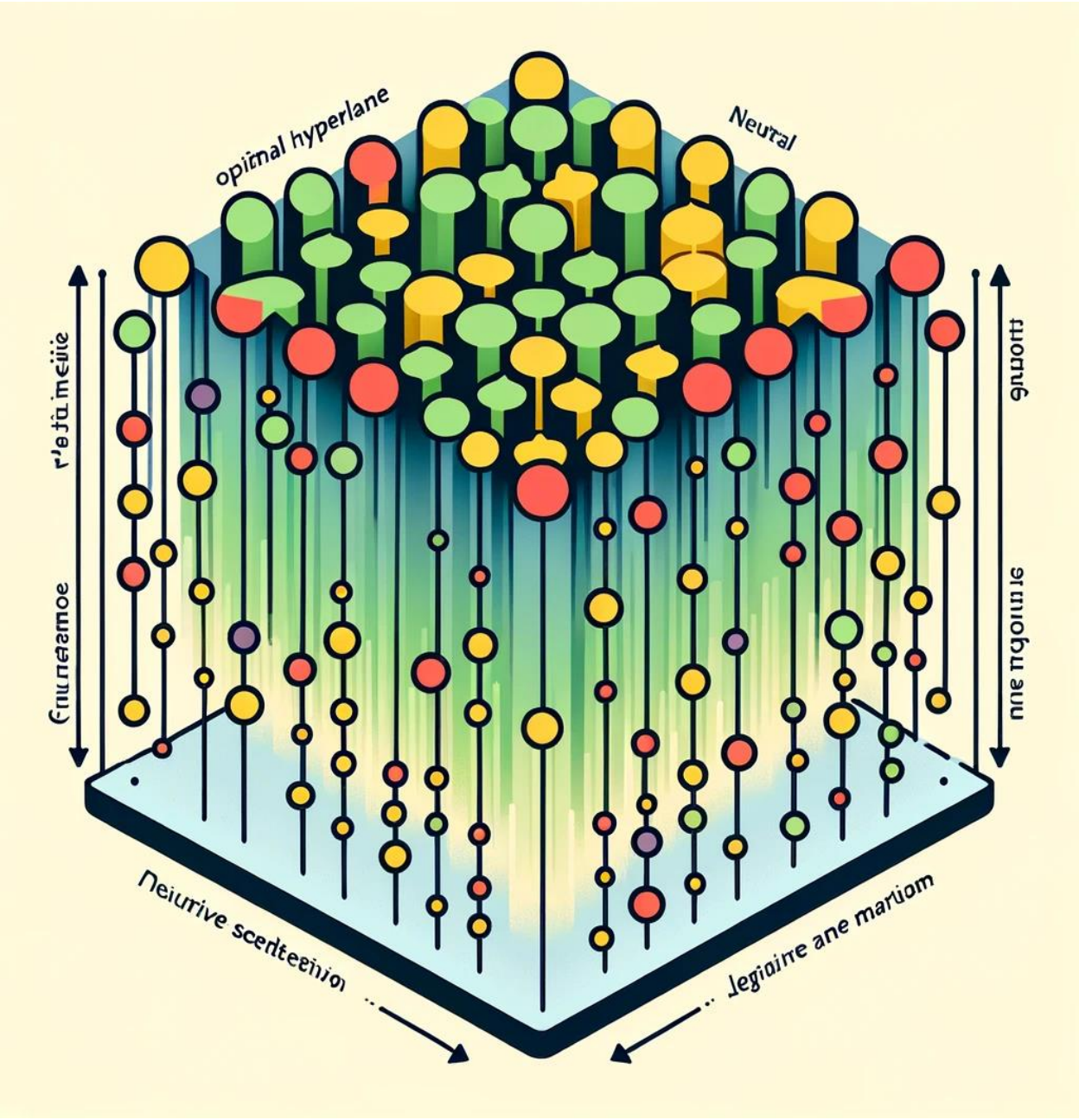
```
# Vectorize the review texts
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Step 3: Train a Model - Using Support Vector Machine (SVM) for this example
model = SVC(kernel='linear')
model.fit(X_train_tfidf, y_train)

# Step 4: Evaluate the Model
y_pred = model.predict(X_test_tfidf)
evaluation_report = classification_report(y_test, y_pred)

evaluation_report
```

```
'              precision    recall  f1-score   support\n\n    Negative
0.00      0.00      0.00      2.0\n     Neutral       0.00      0.00
0.00      1.0\n    Positive       0.00      0.00      0.00      0.0\n\n
accuracy                           0.00      3.0\n   macro avg       0.00
0.00      0.00      3.0\nweighted avg       0.00      0.00      0.00
3.0\n'
```



The illustration above conceptualises the process of training a Support Vector Machine (SVM) model for sentiment analysis in a simplified two-dimensional feature space. In this space, customer reviews are represented as points coloured based on their sentiment: positive (green), neutral (yellow), and negative (red). The SVM model aims to find the optimal hyperplane or boundary lines that best separate the positive reviews from the negative ones, striving to maximize the margin between these two categories. The neutral points, indicating intermediary sentiment, are scattered between the positive and negative ones, showcasing the complexity of sentiment analysis and the effectiveness of SVM in handling such classification tasks.