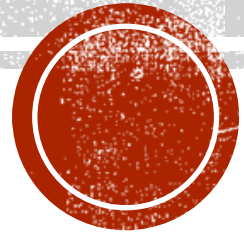
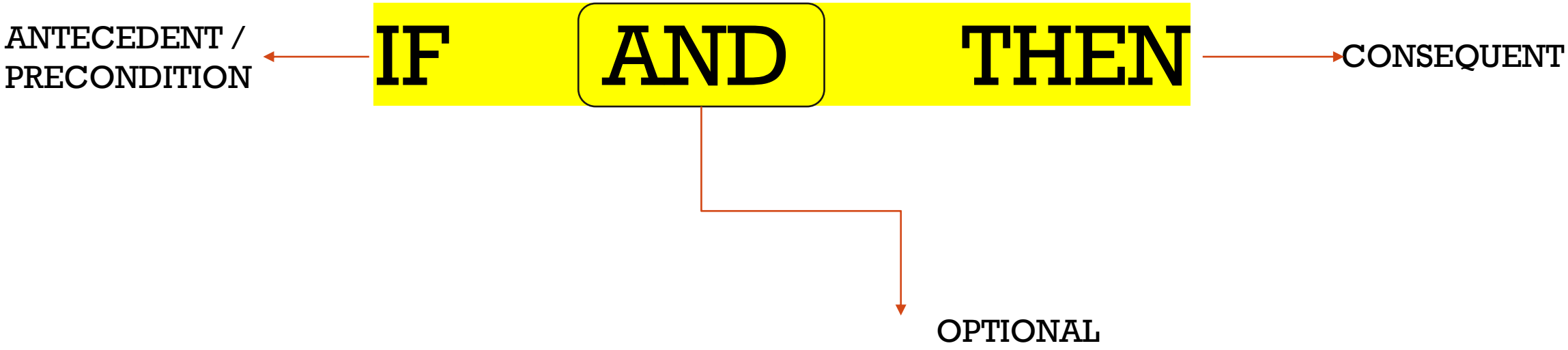
LECTURE # 06

RULE BASED CLASSIFICATION IN KNIME AND PYTHON



- Rule-based classification is a popular and intuitive approach used in machine learning and data mining to classify data based on a set of predefined rules.

COMPONENTS OF RULE BASED CLASSIFICATION



HOW IT WORKS:

Rule-based classification is a part of the supervised learning paradigm,

The classification rules are typically in the form of "if-then" statements, where the antecedent specifies the conditions based on input features, and the consequent indicates the predicted class label.

Rule-based classifiers are often referred to as "if-then" classifiers or decision rule classifiers.

ADVANTAGES OF RULE BASED CLASSIFICATION

Interpretability: Rule-based classifiers produce human-readable rules, which can be easily understood and interpreted by domain experts.

Transparency: The decision-making process of rule-based classifiers is transparent, as the classification is based on explicit rules.

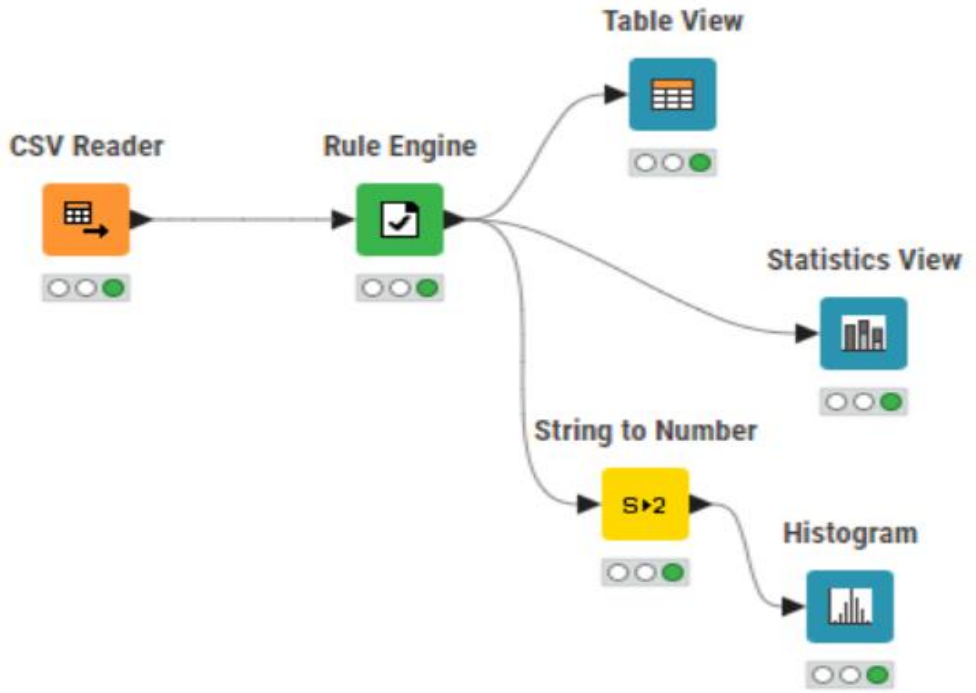
Simplicity: Rule-based classifiers are often simple and easy to implement, making them suitable for problems where complex models are not necessary.

LIMITATIONS OF RULE BASED CLASSIFICATION:

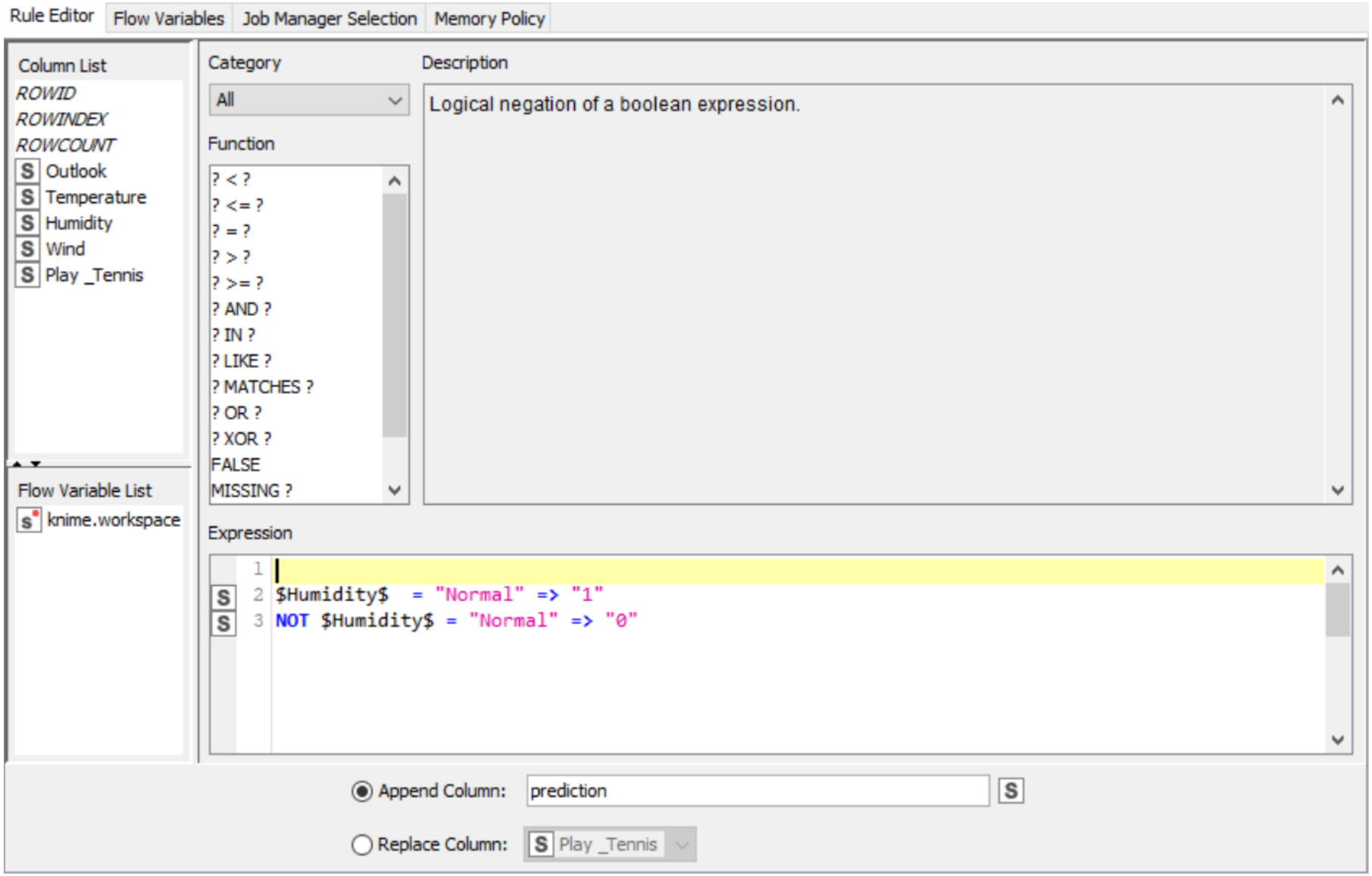
Limited Expressiveness: Rule-based classifiers may struggle to represent complex decision boundaries and capture intricate relationships between features.

Rule Redundancy: In some cases, rule-based classifiers may generate redundant rules, leading to increased complexity and decreased efficiency.

EXAMPLE OF RULE-BASED CLASSIFICATION IN KNIME USING RULE ENGINE:



RULE ENGINE CONFIGURATION:



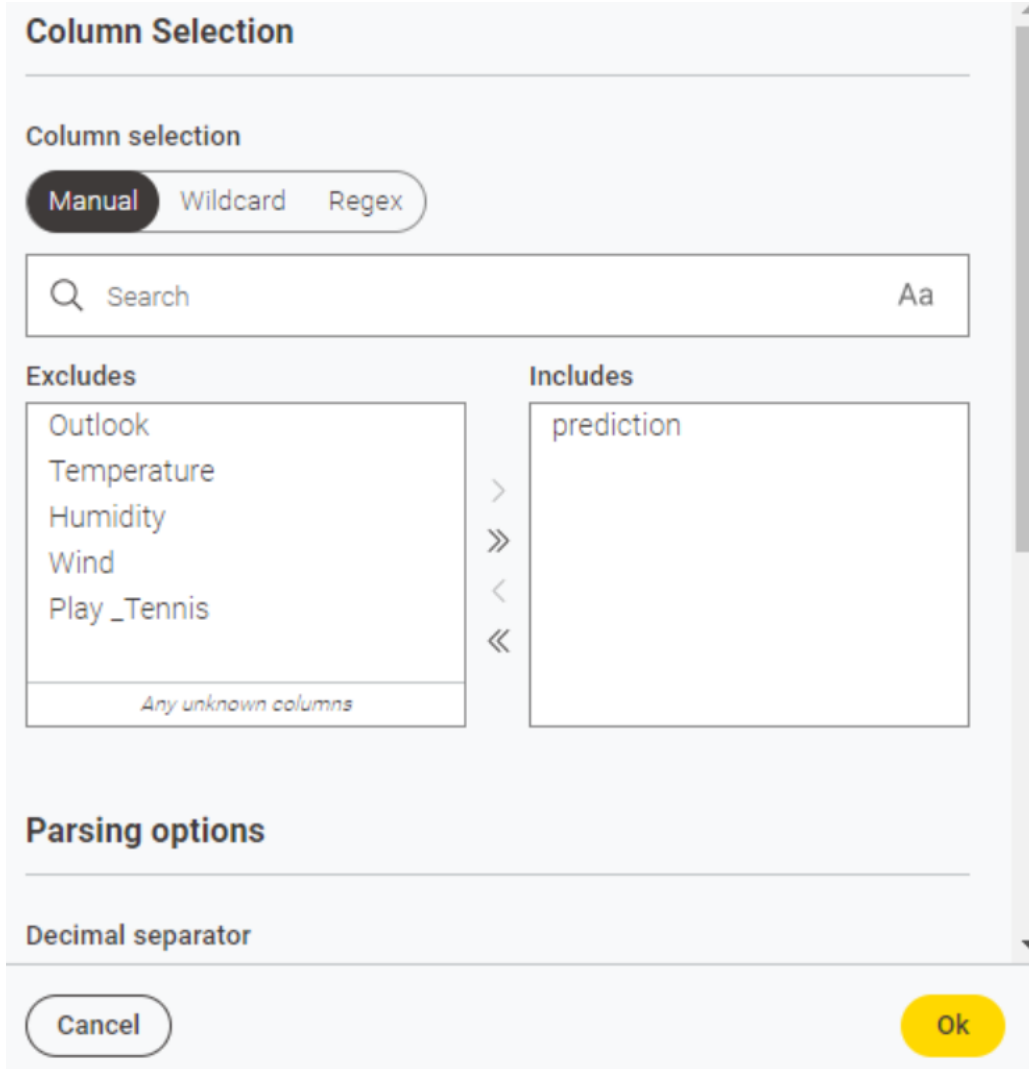
STATISTICS VIEW

Statistics

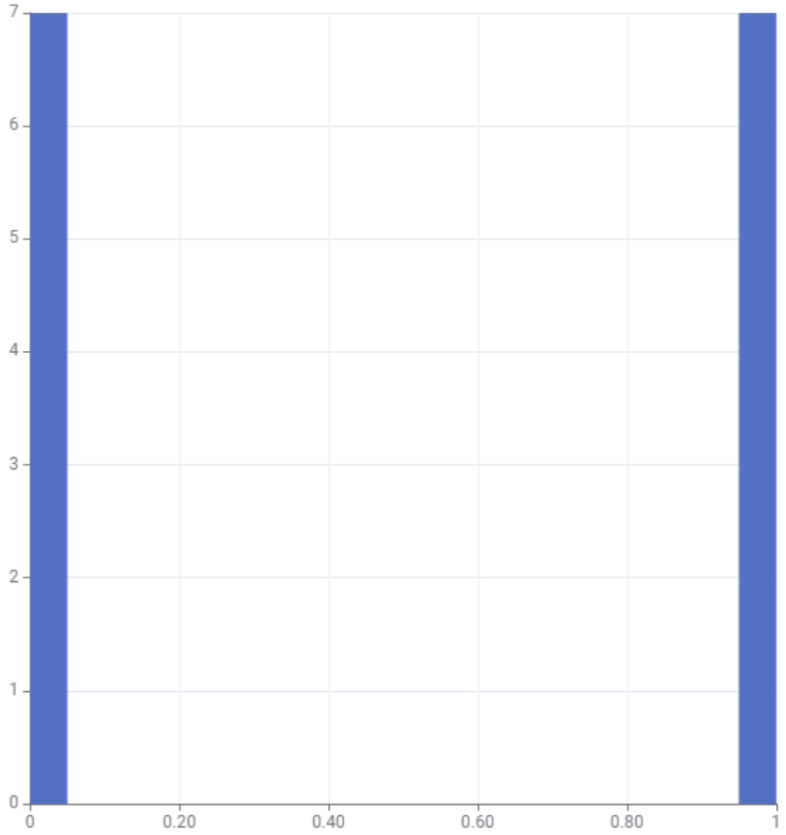
Rows: 6 | Columns: 1

10 most common values
Rain (5; 35.71%), Sunny (5; 35.71%), Overcast (4; 28.57%)
Mild (6; 42.86%), Cool (4; 28.57%), Hot (4; 28.57%)
High (7; 50.0%), Normal (7; 50.0%)
Weak (8; 57.14%), Strong (6; 42.86%)
Yes (8; 57.14%), No (6; 42.86%)
0 (7; 50.0%), 1 (7; 50.0%)

STRING TO NUMBER CONFIGUARTION TO SHOW HISTOGRAM:



HISTOGRAM VIEW (50-50% FOR 0 & 1)




PYTHON EXAMPLES

STEP1 : INSTALL

```
[2] pip install vaderSentiment

Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.0/126.0 kB 1.1 MB/s eta 0:00:00
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2024.2.2)
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

Positive Sentences:

The sun is shining brightly, and it's a beautiful day.

I received a promotion at work, and I'm thrilled about it.

My best friend surprised me with tickets to my favorite concert.

I just adopted a puppy, and I couldn't be happier.

We had a successful project launch, and the team's hard work paid off.

Negative Sentences:

I failed my driving test, and I feel really disappointed.

The restaurant service was terrible, and the food was cold.

My car broke down on the highway, and it's going to cost a fortune to fix.

I got a rejection letter from my dream college, and I'm devastated.

My laptop crashed, and I lost all of my important files.

Neutral Sentences:

The weather forecast predicts a chance of rain later today.

I need to buy groceries and run some errands after work.

The meeting scheduled for tomorrow got postponed to next week.

I'm taking a yoga class to relax and destress after a long day.

The new restaurant in town has mixed reviews, so I'm not sure if I want to try it.

RULE BASED CLASSIFICATION USING VADER SENTIMENT

```
[20] # Importing necessary libraries
     import matplotlib.pyplot as plt
     from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

     # Function to print sentiments and create visualizations
     def analyze_sentiment():
         # Create a SentimentIntensityAnalyzer object
         sid_obj = SentimentIntensityAnalyzer()

         # Input sentence from user
         sentence = input("Enter a sentence: ")

         # polarity_scores method of SentimentIntensityAnalyzer object gives a sentiment dictionary
         sentiment_dict = sid_obj.polarity_scores(sentence)

         print("Overall sentiment dictionary is:", sentiment_dict)
         print("Sentence was rated as {:.2f}% Negative".format(sentiment_dict['neg']*100))
         print("Sentence was rated as {:.2f}% Neutral".format(sentiment_dict['neu']*100))
         print("Sentence was rated as {:.2f}% Positive".format(sentiment_dict['pos']*100))

         print("Sentence Overall Rated As ", end="")

         # Decide sentiment as positive, negative, or neutral
         if sentiment_dict['compound'] >= 0.05:
             print("Positive")
         elif sentiment_dict['compound'] <= -0.05:
             print("Negative")
         else:
             print("Neutral")

         # Create a pie chart for visualization
         labels = ['Positive', 'Neutral', 'Negative']
         sizes = [sentiment_dict['pos'], sentiment_dict['neu'], sentiment_dict['neg']]
         colors = ['green', 'gold', 'red']
         explode = (0.1, 0, 0)  # explode the 1st slice (Positive)
         plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)
         plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
         plt.title('Sentiment Analysis')
         plt.show()

     # Driver code
     if __name__ == "__main__":
         analyze_sentiment()
```

POLARITY SCORE:

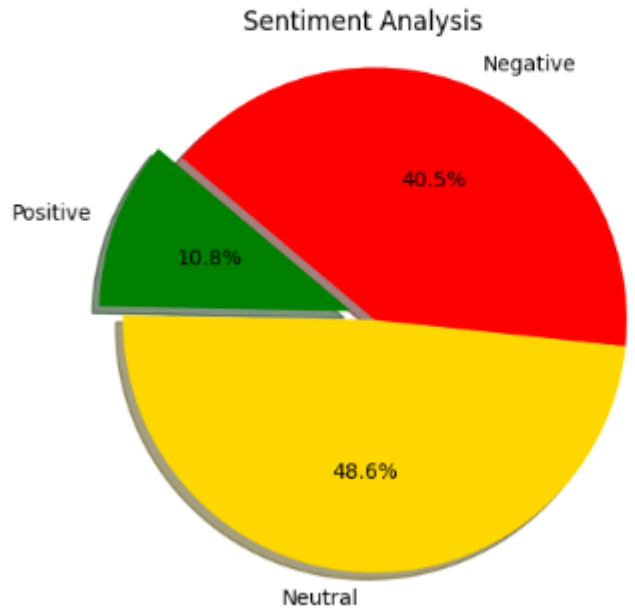
The polarity score, in the context of sentiment analysis, is a measure of the sentiment expressed in a piece of text. It typically ranges from -1 to 1, where:

- Negative values indicate a negative sentiment.
- Positive values indicate a positive sentiment.
- A score of 0 suggests a neutral sentiment.

This score is calculated based on various linguistic features present in the text, such as words, phrases, and emoticons, and how they contribute to the overall sentiment. The polarity score is a quantitative representation of the emotional tone or sentiment conveyed by the text.

OUTPUT:

```
Enter a sentence: I got a rejection letter from my dream college, and I'm devastated.
Overall sentiment dictionary is: {'neg': 0.405, 'neu': 0.486, 'pos': 0.108, 'compound': -0.7579}
Sentence was rated as 40.50% Negative
Sentence was rated as 48.60% Neutral
Sentence was rated as 10.80% Positive
Sentence Overall Rated As Negative
```



RULE BASED CLASSIFICATION USING TEXT BLOB

```
[36] # Importing necessary libraries
     from textblob import TextBlob

     # Function to perform sentiment analysis
     def analyze_sentiment(sentence):
         # Create a TextBlob object
         blob = TextBlob(sentence)

         # Get the sentiment polarity
         sentiment_polarity = blob.sentiment.polarity

         # Decide sentiment based on polarity
         if sentiment_polarity > 0:
             sentiment = "Positive"
         elif sentiment_polarity < 0:
             sentiment = "Negative"
         else:
             sentiment = "Neutral"

         return sentiment

     # Driver code
     if __name__ == "__main__":
         # Input sentence from user
         sentence = input("Enter a sentence: ")

         # Perform sentiment analysis
         sentiment = analyze_sentiment(sentence)

         # Print the result
         print("Sentiment of the sentence '{}' is: {}".format(sentence, sentiment))
```

OUTPUT

```
Enter a sentence: I need to buy groceries and run some errands after work.
Sentiment of the sentence 'I need to buy groceries and run some errands after work.' is: Neutral
```

RULE BASED CLASSIFICATION USING NLTK

```python
# Importing necessary libraries
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')


# Function to perform sentiment analysis
def analyze_sentiment(sentence):
    # Create a SentimentIntensityAnalyzer object
    sid = SentimentIntensityAnalyzer()

    # Get the sentiment score for the sentence
    sentiment_score = sid.polarity_scores(sentence)

    # Decide sentiment based on compound score
    if sentiment_score['compound'] >= 0.05:
        sentiment = "Positive"
    elif sentiment_score['compound'] <= -0.05:
        sentiment = "Negative"
    else:
        sentiment = "Neutral"

    return sentiment

# Driver code
if __name__ == "__main__":
    # Input sentence from user
    sentence = input("Enter a sentence: ")

    # Perform sentiment analysis
    sentiment = analyze_sentiment(sentence)

    # Print the result
    print("Sentiment of the sentence '{}' is: {}".format(sentence, sentiment))
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
Enter a sentence: I need to buy groceries and run some errands after work.
Sentiment of the sentence 'I need to buy groceries and run some errands after work.' is: Neutral
```

TASKS

Task : 01:

Using python implements VADER rules-based classification algorithm to find the sentiments of different sentences.

Task : 02 :Using python implements textBlob rules-based classification algorithm to find the sentiments of different sentences and compare the results with task # 01.