```
from google.colab import drive
drive.mount('/content/drive')
```

```
!pip install gensim pandas scikit-learn openpyxl numpy scipy
!pip install python-docx
```

Import Libraries

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
import gensim.downloader as api
```

Load the Word Embedding Model

```
# Load the pre-trained Word2Vec model
word2vec_model = api.load("glove-wiki-gigaword-50")
```

Load the Datasets

```
# Load the dataset from the provided GitHub link
drug_url = 'https://github.com/Sankaran36/MADM-NB/blob/main/Drug_Attributes_DS.xlsx?raw=true'
drug_df = pd.read_excel(drug_url)

# Load the unstructured text abstract corpus
doc_url = 'https://github.com/Sankaran36/MADM-NB/blob/main/Abstract_Corpus_DS.docx?raw=true'
doc = docx.Document(doc_url)

# Extract text from the document
abstract_corpus = [p.text for p in doc.paragraphs]
```

Feature Vectorization Using Word Embeddings

```
# Define the attributes we are interested in
attributes = ["Efficacy", "SideEffect", "Cost", "Safety"]

# Create a function to get the word embedding for a given attribute
def get_embedding(word):
    try:
        return word2vec_model[word]
    except KeyError:
        # Return a zero vector if the word is not found in the embedding model
        return np.zeros(word2vec_model.vector_size)

# Convert drug attributes to vectors using the word embeddings
drug_vectors = []
for index, row in drug_df.iterrows():
    drug_vector = []
    for attr in attributes:
        embedding = get_embedding(attr.lower())
        drug_vector.extend(embedding)
    drug_vectors.append(drug_vector)

# Convert to numpy array
drug_vectors = np.array(drug_vectors)
```

Data Standardization

```
# Standardize the data
scaler = StandardScaler()
scaled_drug_vectors = scaler.fit_transform(drug_vectors)
```

Principal Component Analysis (PCA)

```
# Apply PCA to reduce dimensionality
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_drug_vectors)
```

Attention Mechanism

```
# Define a simple attention mechanism
def attention_mechanism(query_vector, key_vectors):
    attention_scores = cosine_similarity(query_vector.reshape(1, -1), key_vectors).flatten()
    return attention_scores

# Define a query vector (this should be based on the input symptoms or other criteria)
query_vector = np.mean(principal_components, axis=0)

# Calculate attention scores
attention_scores = attention_mechanism(query_vector, principal_components)
```

Drug Ranking

```
# Add attention scores to the DataFrame
drug_df['AttentionScore'] = attention_scores

# Rank the drugs based on attention scores and other attributes
drug_df['EfficacyRank'] = drug_df['Efficacy'].rank(ascending=False)
drug_df['SafetyRank'] = drug_df['Safety'].rank(ascending=False)
drug_df['CostRank'] = drug_df['Cost'].rank()
drug_df['SideEffectRank'] = drug_df['SideEffect'].rank()

# Final ranking based on the combination of all ranks
drug_df['FinalRank'] = drug_df[['EfficacyRank', 'SafetyRank', 'CostRank', 'SideEffectRank', 'AttentionScore']].mean(axis=1)
ranked_drugs = drug_df.sort_values('FinalRank', ascending=True)

print(ranked_drugs[['Drug', 'FinalRank']])
```

Model Training and Evaluation

```
# For demonstration purposes, we will assume that the first column is the label (Disease) and the rest are features
X = principal_components
y = drug_df['Disease']  # Assuming the label column is named 'Disease'

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
```