DermaGlow AI

Dermatology Essentials and Product Ingredient Analyzer Bot

Group Members

Lat, Betina Grace C.

Montesa, Rosemarie D.

Pino, Renalyn N.

Introduction

Group Members:

Betina Grace C. Lat

Rosemary D. Montesa

Renalyn N. Pino

Introduction

The Dermatology Essentials and Product Ingredient Analyzer Bot helps individuals to make smart choices about skincare essentials. It provide users with a convenient tool to analyze skin concerns and recommend suitable skincare products based on their skin types, specific needs, and ingredient preferences to promote healthier and better skincare habits.

Introduction

Domain

The project is in the dermatology and skincare domain, focusing on analyzing skin concerns and recommending suitable skincare products based on user input. By analyzing these concerns, this bot recommends the right products that would work best for each person, making skincare more personal and effective.

Introduction

Reasoning Method

Natural Language Processing (NLP) for Intent Recognition
This chatbot understands the user's intent and extracts key features (product type, preferences). This is a combination of rule-based and machine learning methods for example, intent classification with Naive Bayes, NN and BERT).

Example:
- The chatbot detects that the user mentioned "*want*" and "*toner,*" so it classifies this intent as a request for a product recommendation.

Implementation:
Combined these with a machine learning classifier trained on labeled intents.

Data Preparation

We collected approved skin care products and other related data from **FDA website** with entries that are timely, significant and complete. To keep things relevant, we selected key fields like product ID, name, type, main ingredients, their benefits, suitability, directions for use, skin concerns, allergens to avoid, pricing, and whether the products are alcohol-free, cruelty-free, or fragrance-free. This way, our dataset stays relevant and meaningful for the analysis.

FDA Website: https://verification.fda.gov.ph/COSMETIC_ALL_INDUSTRYlist.php

Data Preparation

Columns in the Dataset:

- **product_id:** Unique ID for each product.
- **product_name:** Name of the skincare product.
- **product_type:** Category (e.g., Moisturizer, Cleanser).
- **main_ingredients:** Key ingredients in the product.
- **ingredient_benefits:** Benefits of the ingredients.

- **suitability:** Suitable skin types (e.g., Oily, Dry).
- **directions_for_use:** Instructions on how to use the product.
- **skin_concerns:** Common skin issues the product addresses.
- **allergens_ingredients_to_avoid:** Allergens or irritants to avoid.
- **prices**: Product cost.
- Attributes: **alcohol_free, cruelty_free, fragrance_free.**

Data Preparation


```python
✓  Data Collection and Handling

   data = pd.read_csv("/content/derma_essentials-fda.csv")

   data.fillna({
       'product_name': 'Unknown',
       'suitability': 'Unknown',
       'skin_concerns': 'None',
       'main_ingredients': 'None',
       'allergens_ingredients_to_avoid': 'None',
       'prices': 0,
       'cruelty_free': False,
       'fragrance_free': False
   }, inplace=True)

   data = data.drop_duplicates(subset='product_name', keep='first')

   data = data.drop(columns=['Unnamed: 14'])
   print(data.isnull().sum())
```

In this step, we imported the derma_essentials–fda.csv dataset, which contains the necessary data for our analysis.
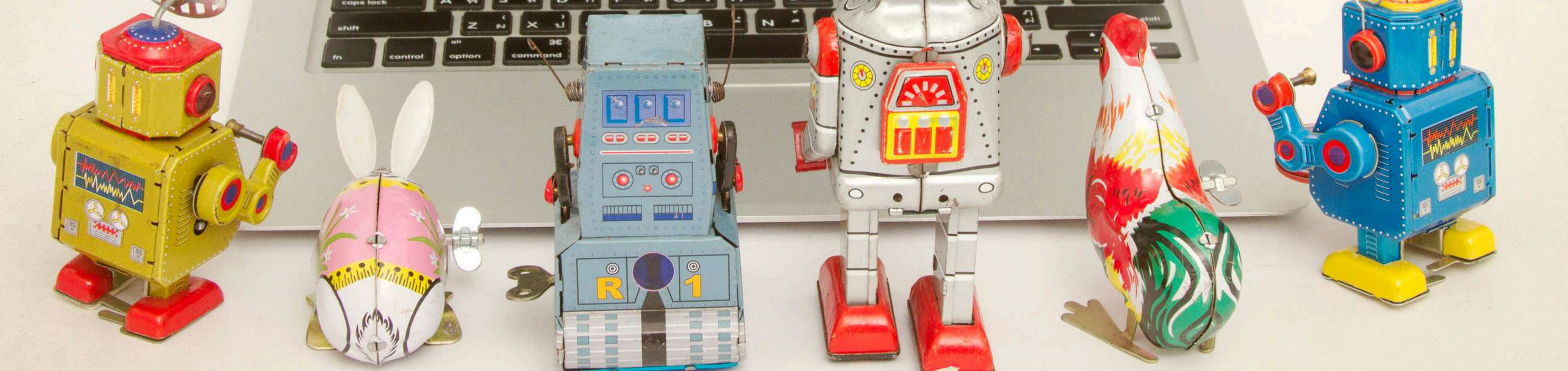
Training the Model

Display training code and explain key parameters (e.g., epochs, learning rate).

Show part of the training output and discuss performance.

Machine Learning Module and Classification Prediction using BERT, Naive Bayes, and Neural Network with TensorFlow

Tokenization Process (BERT)

```python
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
bert_model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(label_mapping))


encoded_labels = [label_mapping[label] for label in labels]

inputs = tokenizer(intents, padding=True, truncation=True, return_tensors="pt")

train_intents, eval_intents, train_labels, eval_labels = train_test_split(
    intents, encoded_labels, test_size=0.2, random_state=42
)
train_inputs = tokenizer(train_intents, padding=True, truncation=True, return_tensors="pt")
eval_inputs = tokenizer(eval_intents, padding=True, truncation=True, return_tensors="pt")
train_labels_tensor = torch.tensor(train_labels)
eval_labels_tensor = torch.tensor(eval_labels)
```

Bert Setup for NLP

We load a pre-trained **BERT model** and **tokenizer** to classify dermatology-related texts, convert dermatology labels like "*acne*" or "*dry*" into numbers, tokenize the texts for BERT while making sure they're padded and truncated, split the data into training and test sets, tokenize both sets separately and convert the labels into tensors for model input.

Naive Bayes

```
[ ]  vectorizer = CountVectorizer()
     X = vectorizer.fit_transform(intents)
     intent_model = MultinomialNB()
     intent_model.fit(X, labels)

     label_encoder = LabelEncoder()
     encoded_labels = label_encoder.fit_transform(labels)

     X_train, X_test, y_train, y_test = train_test_split(X, encoded_labels, test_size=0.2, random_state=42)
```

Naive Bayes Setup

We convert the text data into numerical features with `CountVectorizer`, train the Naive Bayes model using labels like *"acne"* or *"pores,"* encode the labels numerically and split the data into training and test sets for performance evaluation.

Neural Networks using Tensorflow

```python
nn_model = models.Sequential()
nn_model.add(layers.Dense(64, activation='relu', input_dim=X_train.shape[1]))
nn_model.add(layers.Dense(32, activation='relu'))
nn_model.add(layers.Dense(len(set(labels)), activation='softmax'))

nn_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

X_train_dense = X_train.toarray()
X_test_dense = X_test.toarray()
history = nn_model.fit(X_train_dense, y_train, epochs=50, validation_data=(X_test_dense, y_test))

results = nn_model.evaluate(X_test_dense, y_test)

print("Test Loss:", results[0])
print("Test Accuracy:", results[1])

y_pred = nn_model.predict(X_test_dense)
y_pred_classes = tf.argmax(y_pred, axis=1).numpy()

precision = precision_score(y_test, y_pred_classes, average='weighted')
recall = recall_score(y_test, y_pred_classes, average='weighted')

print("Test Precision:", precision)
print("Test Recall:", recall)

results = nn_model.evaluate(X_test_dense, y_test)
```

Neural Network Setup with TensorFlow

The code sets up a neural network with two hidden layers and uses softmax for multi-class classification. After training, it checks the model's performance, makes predictions, and calculates precision and recall.

Testing and Evaluation

- Show the chatbot in action with test cases.

- Discuss performance metrics like accuracy, precision, and recall

Training and Validation Accuracy & Loss
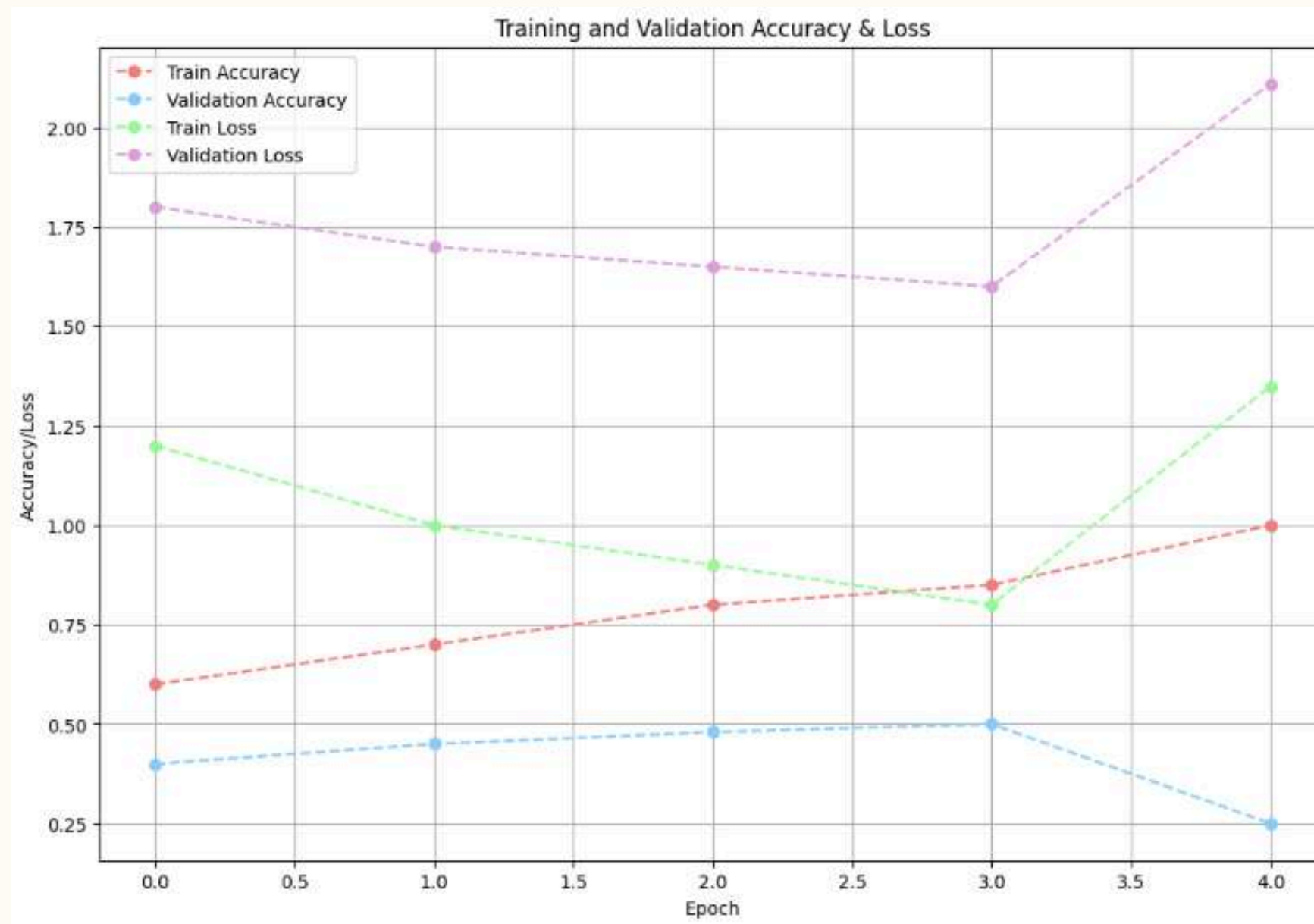
Results

Evaluation Results

- The training process over 50 epochs showed improvements in training performance, with accuracy rising from 86.67% to **93.33%** and loss decreasing from **0.91 to 0.78**. By the end of the 50th epoch, this increased to 93.33%, showing that the model became better at recognizing patterns and making correct predictions during training.

Evaluation Results

50 Epochs Training Cycles


```
Epoch 20/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step - accuracy: 0.8667 - loss: 1.4723 - val_accuracy: 0.0000e+00 - val_loss: 1.9661
Epoch 21/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step - accuracy: 0.8667 - loss: 1.4458 - val_accuracy: 0.0000e+00 - val_loss: 1.9605
Epoch 22/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 122ms/step - accuracy: 0.8667 - loss: 1.4198 - val_accuracy: 0.0000e+00 - val_loss: 1.9548
Epoch 23/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 284ms/step - accuracy: 0.8667 - loss: 1.3937 - val_accuracy: 0.0000e+00 - val_loss: 1.9491
Epoch 24/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - accuracy: 0.8667 - loss: 1.3675 - val_accuracy: 0.0000e+00 - val_loss: 1.9431
Epoch 25/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - accuracy: 0.8667 - loss: 1.3415 - val_accuracy: 0.0000e+00 - val_loss: 1.9365
Epoch 26/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step - accuracy: 0.8667 - loss: 1.3151 - val_accuracy: 0.2500 - val_loss: 1.9301
Epoch 27/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9333 - loss: 1.2887 - val_accuracy: 0.2500 - val_loss: 1.9239
Epoch 28/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - accuracy: 0.9333 - loss: 1.2625 - val_accuracy: 0.2500 - val_loss: 1.9179
Epoch 29/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 167ms/step - accuracy: 0.9333 - loss: 1.2365 - val_accuracy: 0.5000 - val_loss: 1.9125
Epoch 30/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 201ms/step - accuracy: 0.9333 - loss: 1.2105 - val_accuracy: 0.5000 - val_loss: 1.9079
```

```
Epoch 40/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 147ms/step - accuracy: 0.9333 - loss: 0.9564 - val_accuracy: 0.5000 - val_loss: 1.8722
Epoch 41/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step - accuracy: 0.9333 - loss: 0.9319 - val_accuracy: 0.5000 - val_loss: 1.8692
Epoch 42/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 150ms/step - accuracy: 0.9333 - loss: 0.9077 - val_accuracy: 0.5000 - val_loss: 1.8664
Epoch 43/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 315ms/step - accuracy: 0.9333 - loss: 0.8840 - val_accuracy: 0.5000 - val_loss: 1.8638
Epoch 44/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 273ms/step - accuracy: 0.9333 - loss: 0.8606 - val_accuracy: 0.5000 - val_loss: 1.8618
Epoch 45/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step - accuracy: 0.9333 - loss: 0.8378 - val_accuracy: 0.5000 - val_loss: 1.8598
Epoch 46/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - accuracy: 0.9333 - loss: 0.8154 - val_accuracy: 0.5000 - val_loss: 1.8581
Epoch 47/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 154ms/step - accuracy: 0.9333 - loss: 0.7937 - val_accuracy: 0.5000 - val_loss: 1.8563
Epoch 48/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 0.9333 - loss: 0.7724 - val_accuracy: 0.5000 - val_loss: 1.8546
Epoch 49/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 126ms/step - accuracy: 0.9333 - loss: 0.7516 - val_accuracy: 0.5000 - val_loss: 1.8529
Epoch 50/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 196ms/step - accuracy: 0.9333 - loss: 0.7313 - val_accuracy: 0.5000 - val_loss: 1.8511
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.5000 - loss: 1.8511
Test Loss: 1.8511312007904053
Test Accuracy: 0.5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 233ms/step
Test Precision: 0.625
Test Recall: 0.5
```

We trained the model for 50 epochs due to the limited size of our dataset.

Evaluation Results

Training vs Validation/Test Accuracy:

- The training accuracy is relatively high (**93.33%**), and the test accuracy is **50%**. This indicates that the model is performing better in terms of generalizing to unseen data compared to previous runs where the test accuracy was lower. There is still a bit of overfitting where training accuracy is higher than test accuracy.

Precision and Recall:

- Test precision is 0.625, which means that when the model predicts a positive class, it is correct **62.5%** of the time. Test recall is 0.5, meaning the model correctly identifies **50%** of all true positive instances. These metrics suggest that the model is making reasonably accurate predictions, with room for improvement in both precision and recall.

Challenges and Future Suggestions

Challenges

- High training accuracy (93.33%) but low test accuracy (50%), indicating overfitting.
- Test precision (0.625) and recall (0.5) show the model can improve in identifying true positives.

Suggestions for Improvement:

- Increase the dataset size to help our model generalize better.
- Apply regularization techniques to reduce overfitting.
- Experiment with different algorithms to improve precision and recall.

Conclusion

This **Dermatology Essential and Product Analyzer Bot** performed well during operation. It can answer some skin care questions questions accurately including all the features that we added. The model has a high training accuracy of **93.33%**, but its test accuracy drops to **50%**, likely due to small dataset, leading to overfitting. It correctly predicts **62.5% of positive cases** (precision) but only identifies **50% of all true positives** (recall). This means the model is fairly accurate but could still improve in identifying more positive cases. Future suggestions include increasing the dataset size, trying regularization techniques to reduce overfitting, as well as experimenting with different algorithms to improve both precision and recall.

References

DATASET SOURCE

FDA Website: https://verification.fda.gov.ph/COSMETIC_ALL_INDUSTRYlist.php

LEARNING PACKET TOPICS 8-10
https://classroom.google.com/c/NzA4NTE1Mjk2MjMz/m/NzM5MzAyNjQ1OTk5/details

SUMMARY TOPIC 1-6
https://classroom.google.com/c/NzA4NTE1Mjk2MjMz/m/NzM5MzAyNjQ1OTk5/details

Thanks for watching...