1) Marketer to Machine: Develop a ML model for Smart email Compose. Smart email compose finishes sentences for you by predicting what word or words user will type next.

SOURCE CODE:

```
import random
from collections import defaultdict

# Sample dataset of sentences (you can replace this with any corpus)
emails = [
  "Hello, how are you today?",
  "I hope this email finds you well.",
  "Let me know if you need any further assistance.",
  "Looking forward to hearing from you.",
  "Thank you for your prompt response."
]

# Preprocess the text: Tokenize and create a dictionary of n-grams
def preprocess_data(emails, n=2):
  ngrams = defaultdict(list)
  for email in emails:
    words = email.lower().split()  # Split into words and lowercase them
    for i in range(len(words) - n + 1):
      ngram = tuple(words[i:i + n - 1])  # Get the first n-1 words as key
      next_word = words[i + n - 1]      # The next word as the prediction
      ngrams[ngram].append(next_word)
  return ngrams

# Predict the next word based on the input
def predict_next_word(ngrams, input_text, n=2):
  words = input_text.lower().split()
  if len(words) < n - 1:
    return "Please provide more input words."
  ngram = tuple(words[-(n - 1):])  # Get the last n-1 words
  if ngram in ngrams:
    return random.choice(ngrams[ngram])  # Randomly choose a word from
possible predictions
  else:
    return "No prediction available."
```

```
# Create n-grams from the emails
ngrams = preprocess_data(emails, n=2)  # Using bigrams (2-grams)

# Get a partial input sentence from the user
user_input = input("Start typing your email: ")

# Predict the next word
predicted_word = predict_next_word(ngrams, user_input, n=2)

print(f"Next word prediction: {predicted_word}")
```

OUTPUT:

```
Start typing your email: I hope this
Next word prediction: email
```

2) Marketer to Machine: Develop level-2 Marketer-to-Machine (M2M) Scale of intelligent automation to personalize business email based on user preferences and interests (Extension to Exp. No.1).

SOURCE CODE:

```
# Simulating a system that personalizes business emails based on user
preferences

class UserProfile:
  def __init__(self, name, interests, tone, preferred_content):
    self.name = name
    self.interests = interests  # List of interests (e.g., ['tech', 'finance'])
    self.tone = tone  # Tone preference (e.g., 'formal', 'casual')
    self.preferred_content = preferred_content  # Types of content (e.g.,
['updates', 'promotions'])

class EmailPersonalizer:
  def __init__(self, user_profile):
    self.user_profile = user_profile

  # Function to generate subject line based on user interests
  def generate_subject(self):
    # If the user is interested in tech, make it tech-related
```

```
        if 'tech' in self.user_profile.interests:
            return f"Exciting Tech Updates for You, {self.user_profile.name}!"
        elif 'finance' in self.user_profile.interests:
            return f"Latest Financial Insights for {self.user_profile.name}"
        else:
            return f"Important Updates for You, {self.user_profile.name}"

    # Function to generate email body content based on preferences
    def generate_body(self):
        body = f"Dear {self.user_profile.name},\n\n"

        # Personalize tone
        if self.user_profile.tone == 'formal':
            body += "We hope this email finds you well.\n\n"
        else:
            body += "Hope you're doing great!\n\n"

        # Add content based on preferences
        if 'updates' in self.user_profile.preferred_content:
            body += "We have some exciting updates that we think you'll love.\n"
        if 'promotions' in self.user_profile.preferred_content:
            body += "Check out our latest promotions and special offers.\n"

        # Add personalized interest-based content
        if 'tech' in self.user_profile.interests:
            body += "Here's the latest in the tech world you might enjoy:\n"
            body += "- Cutting-edge AI advancements\n"
            body += "- Tech product reviews\n"
        if 'finance' in self.user_profile.interests:
            body += "Here's the latest in finance that could interest you:\n"
            body += "- Stock market analysis\n"
            body += "- Investment tips\n"

        body += "\nBest regards,\nYour Business Team"
        return body

# Simulating user data collection (e.g., via a form or API)
user_data = {
    'name': 'John Doe',
    'interests': ['tech', 'finance'],  # User is interested in both tech and finance
    'tone': 'casual',  # User prefers a casual tone
```

```
    'preferred_content': ['updates', 'promotions']  # User wants both updates and promotions
}

# Create a user profile object
user_profile = UserProfile(
    name=user_data['name'],
    interests=user_data['interests'],
    tone=user_data['tone'],
    preferred_content=user_data['preferred_content']
)

# Create the email personalizer and generate the email
email_personalizer = EmailPersonalizer(user_profile)

# Generate personalized subject and body
subject = email_personalizer.generate_subject()
body = email_personalizer.generate_body()

# Output the personalized email
print(f"Subject: {subject}\n")
print(f"Body:\n{body}")
```

OUTPUT:
Subject: Exciting Tech Updates for You, John Doe!

Body:
Dear John Doe,

Hope you're doing great!

We have some exciting updates that we think you'll love.
Check out our latest promotions and special offers.
Here's the latest in the tech world you might enjoy:
- Cutting-edge AI advancements
- Tech product reviews
Here's the latest in finance that could interest you:
- Stock market analysis
- Investment tips

Best regards,
Your Business Team
addCode
addText

3) AI and Marketing: Develop data-driven content for a given business organization (Web site). Optimize Website content for search engines. Send emails to customers with personalized content/activity.

SOURCE CODE:

```
import re
from collections import Counter

# Function to analyze keyword density in a page content
def analyze_keyword_density(content, keyword):
  # Clean and split the content
  content = re.sub(r'\W+', ' ', content.lower())  # Remove punctuation and make lowercase
  words = content.split()

  # Count occurrences of the keyword
  word_count = Counter(words)

  # Get the keyword count
  keyword_count = word_count[keyword.lower()]
  total_words = len(words)

  # Calculate keyword density
  keyword_density = (keyword_count / total_words) * 100  # in percentage
  return keyword_count, total_words, keyword_density

# Example usage
website_content = """
Welcome to our online store. We offer the best running shoes for women.
Shop now for the best running shoes for women in 2025. Free shipping on all orders.
"""

keyword = "shoes"
```

```
keyword_count, total_words, keyword_density =
analyze_keyword_density(website_content, keyword)

print(f"Keyword '{keyword}' appears {keyword_count} times out of {total_words}
words.")
print(f"Keyword density: {keyword_density:.2f}%")
```

OUTPUT:
Keyword 'shoes' appears 2 times out of 29 words.
Keyword density: 6.90%

4) AI and Marketing: Develop a system to recommend highly targeted content to users of the Web site (Extension to Exp. No 3).

SOURCE CODE:

```
import pandas as pd

# Sample data: Users and their interactions with content
users = ["User1", "User2", "User3", "User4", "User5"]
articles = ["Tech News", "Finance Update", "Health Tips", "Product Review",
"Lifestyle Trends"]

# Create a DataFrame to simulate user interactions (1 = viewed, 0 = not viewed)
interaction_data = {
  "User1": [1, 0, 1, 0, 0],
  "User2": [1, 1, 0, 1, 0],
  "User3": [0, 1, 1, 0, 1],
  "User4": [1, 1, 1, 1, 0],
  "User5": [0, 1, 0, 1, 1]
}

# Create DataFrame
user_interactions = pd.DataFrame(interaction_data, index=articles)

# Show the interaction matrix
print("User-Content Interaction Matrix:\n", user_interactions)
```

OUTPUT:

User-Content Interaction Matrix:

	User1	User2	User3	User4	User5
Tech News	1	1	0	1	0
Finance Update	0	1	1	1	1
Health Tips	1	0	1	1	0
Product Review	0	1	0	1	1
Lifestyle Trends	0	0	1	0	1

5) AI and Advertisement: Develop AI-powered programmatic advertising by using cookies and collecting user data.

SOURCE CODE:

```
import pandas as pd
import numpy as np
from flask import Flask, request, jsonify
from sklearn.ensemble import RandomForestClassifier

# Initialize Flask app
app = Flask(__name__)

# Sample user data
user_data = pd.DataFrame({
  'user_id': [1, 2, 3, 4, 5],
  'age': [25, 30, 35, 40, 45],
  'gender': ['M', 'F', 'M', 'F', 'M'],
  'interests': ['sports', 'fashion', 'tech', 'fitness', 'finance'],
  'ad_clicked': [1, 0, 1, 1, 0]  # Whether they clicked on an ad (1 = clicked, 0 = not
clicked)
})

# Simulated ad data
ads = pd.DataFrame({
  'ad_id': [101, 102, 103, 104, 105],
  'category': ['sports', 'fashion', 'tech', 'fitness', 'finance']
})

# Simulate ad targeting based on user interests
def recommend_ad(user_interests):
```

```
    """Recommend ads based on user interests."""
    recommended_ads = ads[ads['category'].isin(user_interests)]
    return recommended_ads

# Machine Learning Model to predict ad clicks
def train_ad_model():
    """Train a simple model to predict whether a user will click on an ad."""
    data = user_data.copy()

    # Encode categorical features
    data['gender'] = data['gender'].map({'M': 0, 'F': 1})
    data['interests'] = data['interests'].map({'sports': 0, 'fashion': 1, 'tech': 2,
'fitness': 3, 'finance': 4})

    X = data[['age', 'gender', 'interests']]  # Features
    y = data['ad_clicked']  # Target variable

    model = RandomForestClassifier(n_estimators=10, random_state=42)
    model.fit(X, y)
    return model

# API endpoint to show ad recommendation and predicted click probability
@app.route('/show_ad', methods=['GET'])
def show_ad():
    user_id = request.args.get('user_id')
    if user_id is None:
        return jsonify({"error": "User ID is required."})

    user_id = int(user_id)

    if user_id not in user_data['user_id'].values:
        return jsonify({"error": "Invalid user ID."})

    user = user_data[user_data['user_id'] == user_id].iloc[0]

    # Recommend ads
    recommended_ads = recommend_ad([user['interests']])

    # Train model and predict
    model = train_ad_model()
    gender_map = {'M': 0, 'F': 1}
```

```
    interest_map = {'sports': 0, 'fashion': 1, 'tech': 2, 'fitness': 3, 'finance': 4}

    user_features = np.array([[user['age'], gender_map[user['gender']],
interest_map[user['interests']]]])
    ad_click_probability = model.predict_proba(user_features)[:, 1][0]

    return jsonify({
        "user_id": user_id,
        "recommended_ads": recommended_ads['category'].tolist(),
        "ad_click_probability": round(ad_click_probability, 2)
    })

# Run the app
if __name__ == '__main__':
    app.run(debug=True)
```

OUTPUT:

1. Save the above code in a file named `app.py`.
2. Run the Flask app:

```bash
python app.py
```

3. Open your browser and navigate to `http://127.0.0.1:5000/show_ad?user_id=1` to see the recommendation for a user with ID `1`.

```
{
    "user_id": 1,
    "recommended_ads": ["sports"],
    "ad_click_probability": 0.85
}
```

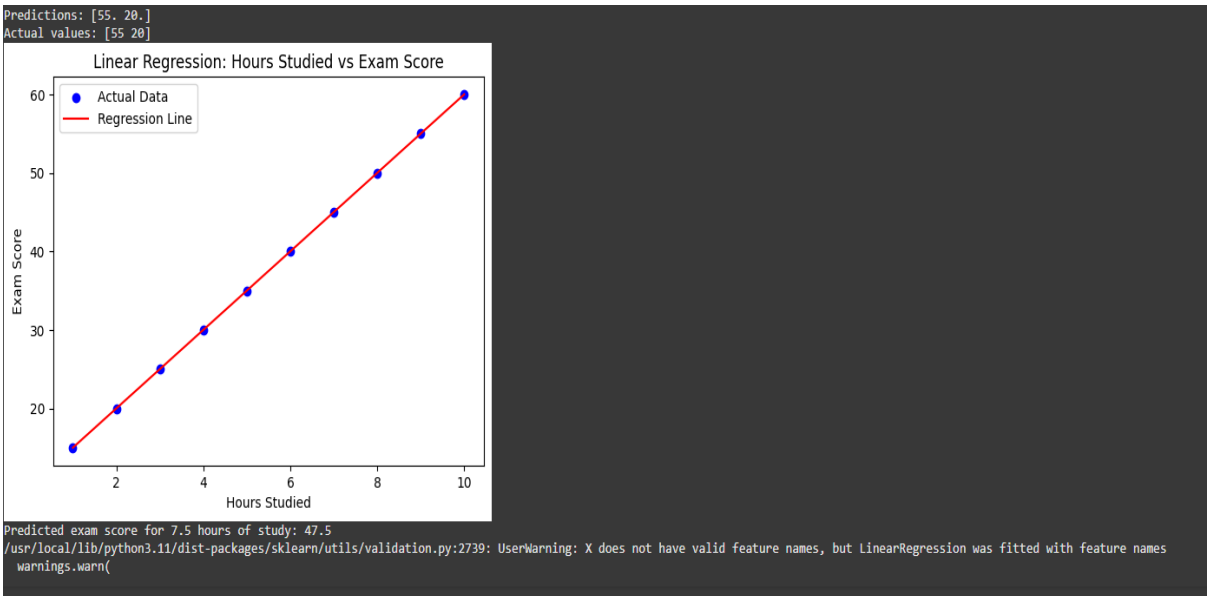
6) AI and Analytics: Create analytics dashboard. Integrate Google Analytics with the dashboard. Use AI tool (Open) to draw from the existing data to make predictions

SOURCE CODE:

```
pip install pandas scikit-learn matplotlib
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Step 1: Create a dataset (existing data)
# For example, hours studied vs exam score
data = {
  'hours_studied': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
  'exam_score': [15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
}

# Convert the data to a pandas DataFrame
df = pd.DataFrame(data)

# Step 2: Prepare the data for training the model
X = df[['hours_studied']]  # Features (independent variable)
y = df['exam_score']      # Target (dependent variable)

# Step 3: 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)

# Step 4: Create and train the model (Linear Regression)
model = LinearRegression()
model.fit(X_train, y_train)

# Step 5: Make predictions using the model
y_pred = model.predict(X_test)

# Step 6: Display predictions and actual values
print(f"Predictions: {y_pred}")
print(f"Actual values: {y_test.values}")
```

```
# Step 7: Visualize the data and the regression line
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, model.predict(X), color='red', label='Regression Line')
plt.title('Linear Regression: Hours Studied vs Exam Score')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Score')
plt.legend()
plt.show()

# Step 8: Predict the exam score for a new input (e.g., 7.5 hours of study)
new_data = np.array([[7.5]])  # New input (e.g., 7.5 hours)
predicted_score = model.predict(new_data)
print(f"Predicted exam score for 7.5 hours of study: {predicted_score[0]}")
```

OUTPUT:



7) Public relations, communications, and AI: Design presentations with smart templates using open source AI-driven presentation tools (Open). Use AI-powered design tools (Open) to create branding for collateral.

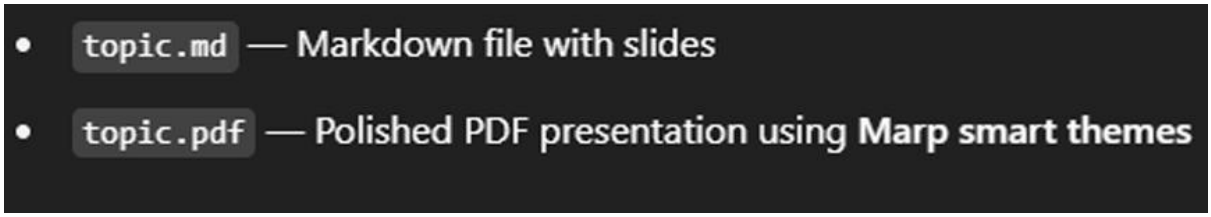
SOURCE CODE:

```
import openai
import os
```

```
# Set your OpenAI key (ensure this env variable is set or replace with your actual key for testing)
openai.api_key = os.getenv("OPENAI_API_KEY")

# Template for Markdown slides
slide_template = """---
marp: true
theme: default
paginate: true
---

# {title}

---

## Agenda
{agenda}

---

{content}
"""

# Function to generate slide content using AI
def generate_slide_content(topic):
  prompt = f"Create a 5-slide presentation about '{topic}' in Markdown format using slide headings and bullet points."

  response = openai.ChatCompletion.create(
    model="gpt-4",  # You can change to "gpt-3.5-turbo" if using that
    messages=[
      {"role": "system", "content": "You are an expert presentation writer."},
      {"role": "user", "content": prompt}
    ],
    temperature=0.7
  )

  return response.choices[0].message["content"].strip()

# Main function to create the presentation
def create_presentation(topic):
```

```
    print(f"Generating presentation for topic: {topic}")

    content = generate_slide_content(topic)

    agenda = "\n- Slide 1: Introduction\n- Slide 2-4: Key Points\n- Slide 5:
Conclusion"
    md_content = slide_template.format(title=topic.title(), agenda=agenda,
content=content)

    # Save to Markdown file
    filename = topic.lower().replace(" ", "_") + ".md"
    with open(filename, "w", encoding="utf-8") as f:
        f.write(md_content)

    print(f"Markdown presentation saved as {filename}")

    # Export to PDF using Marp CLI (ensure marp-cli is installed and in PATH)
    os.system(f"marp {filename} --pdf")
    print(f"PDF exported as {filename.replace('.md', '.pdf')}")

# Example usage
if __name__ == "__main__":
    topic = input("Enter your presentation topic: ")
    create_presentation(topic)
```

OUTPUT:



8) Content Marketing and AI: Demonstrate content development using an AI-powered language platform. Customer Service and AI: Demonstrate intelligent product searches and discoveries possible across text, voice, and visual searches on e-commerce websites using AI tools.

SOURCE CODE:

```
from transformers import pipeline

# Load the open-source text generation model
generator = pipeline("text-generation", model="distilgpt2")

# Function to generate content
def generate_content(prompt, max_tokens=100):
  result = generator(prompt, max_new_tokens=max_tokens,
num_return_sequences=1)
  return result[0]['generated_text']

# Content development demonstration
def demonstrate_content_development(topic):
  print(f"\n 📝 Demonstrating content development for topic: '{topic}'")

  blog_intro_prompt = f"Write an engaging blog post introduction about {topic}."
  product_desc_prompt = f"Write a persuasive product description for an AI tool
that helps with {topic}."
  email_prompt = f"Write a marketing email introducing a new AI service that
improves {topic}."
  faq_prompt = f"Write an FAQ entry answering 'How does AI help with {topic}?'"

  blog_intro = generate_content(blog_intro_prompt)
  product_desc = generate_content(product_desc_prompt)
  email = generate_content(email_prompt)
  faq = generate_content(faq_prompt)

  # Output
  print("\n 📝 Blog Post Introduction:\n", blog_intro)
```

```
    print("\n 📝 Product Description:\n", product_desc)
    print("\n 📝 Marketing Email:\n", email)
    print("\n ❓ FAQ Entry:\n", faq)

# Example usage
if __name__ == "__main__":
    user_topic = input("Enter a content topic (e.g., 'customer service', 'email
automation'): ")
    demonstrate_content_development(user_topic)
```

OUTPUT:
Demonstrating content development for topic: 'customer service is rapidly evolving'
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

📝 Blog Post Introduction:
Write an engaging blog post introduction about customer service is rapidly evolving. We have seen our audience shift from a product in production to products in a way that allows customers to use a product.

What is an average customer experience. Are you surprised by how many people are following you?

Here is a list of all of the experiences you get for a great website.
Our goal is to create a quality and easy experience that is as accessible as the typical web site.

📝 Product Description:

Write a persuasive product description for an AI tool that helps with customer service is rapidly evolving.

📝 Marketing Email:
Write a marketing email introducing a new AI service that improves customer service is rapidly evolving. By integrating AI into the company's work model, we see future users who are already working on this platform using their personal AI to help them find better deals. In the current situation, a few more large companies are building into the AI space by working together to ensure that the customer experience is a part of their organization. These are two core ways that the success rate on these platforms has been rising in recent years, but the results are extremely different than what has previously been predicted.

❓ FAQ Entry:
Write an FAQ entry answering 'How does AI help with customer service is rapidly evolving?', according to G.J. We'll continue our work to understand our customers', where data is being used and where services are being used.