Предложения за Дипломни Проекти

Специалност: Програмиране на Изкуствен Интелект (12. клас)

Организация на защитата:

• **Теория:** Представяне на концепции, алгоритми и презентация.

• **Практика:** Демонстрация на работещ софтуер.

• **Среда за изпълнение:** Кодовете са подготвени за бързо и лесно стартиране в **Google Colab** (с изключение на уеб камера проектите, които се изпълняват локално в среда като VS Code/ PyCharm).

Категория 1: Компютърно зрение (Computer Vision)

1. Разпознаване на пътни знаци за автономни автомобили

Теория: Архитектура и принципи на работа на Конволюционните невронни мрежи (CNN). Оптимизация на хиперпараметри.

Практика: Обучение на CNN модел, класифициращ пътни знаци. (За симулация се използват синтетични данни).

Сорс код (Python / TensorFlow):

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np

X_train = np.random.rand(100, 32, 32, 3)
y_train = np.random.randint(0, 43, size=(100,))

model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,
3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(43, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=3, batch_size=32)
model.save('traffic_sign_model.h5')
print("Моделът е записан успешно!")
```

2. Система за контрол на достъпа чрез лицево разпознаване

Теория: Биометрични данни и алгоритми за откриване и разпознаване на лица (Haar Cascades, MTCNN). Етични аспекти.

Практика: Локално приложение с уеб камера, очертаващо лица и симулиращо разрешаване на достъп.

Сорс код (Python / OpenCV - Изпълнява се локално):

```
import cv2

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret: break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30))

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame, "ACCESS GRANTED", (x, y-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)

    cv2.imshow('Face Recognition Access Control', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'): break

cap.release()
cv2.destroyAllWindows()
```

3. Автоматично класифициране на медицински изображения

Теория: Концепцията за Transfer Learning (пренос на обучение) чрез предобучени мрежи (ResNet, MobileNet).

Практика: Класификация на рентгенови снимки за засичане на аномалии (пневмония).

Сорс код (Python / Transfer Learning):

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras import layers, models
import numpy as np

base_model = MobileNetV2(input_shape=(224, 224, 3), include_top=False,
weights='imagenet')
base_model.trainable = False

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
fake_xrays = np.random.rand(10, 224, 224, 3)
predictions = model.predict(fake_xrays)
print("Прогнози за тест на изображенията:
", predictions)
```

4. Проследяване на обекти и броене на трафик в реално време

Теория: Алгоритми за засичане на обекти в реално време (YOLO - You Only Look Once). Принципи на single-shot детекторите.

Практика: Използване на YOLOv8 за разпознаване на превозни средства и хора от видео поток или камера.

Сорс код (Python / Ultralytics YOLOv8):

```
import cv2
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
cap = cv2.VideoCapture(0) # Може да се замени с 'traffic.mp4'

while cap.isOpened():
    success, frame = cap.read()
    if not success: break
    results = model(frame, verbose=False)
    annotated_frame = results[0].plot()
    cv2.imshow("YOLOv8 Traffic Detection", annotated_frame)
    if cv2.waitKey(1) & 0xFF == ord('q'): break

cap.release()
cv2.destroyAllWindows()
```

Категория 2: Обработка на естествен език (NLP) и Текстов анализ

5. Смарт асистент (Чатбот) за клиентска поддръжка

Теория: Обработка на естествен език (NLP). Разпознаване на намерения (Intent Recognition) и извличане на същности.

Практика: TF-IDF векторизация и изчисляване на косинусово сходство за автоматични отговори на въпроси.

Сорс код (Python / Scikit-learn):

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

faq = {
    "Здравейте, кога ще пристигне поръчката ми?": "Стандартната доставка отнема от 2 до 4 работни дни.",
    "Какви са начините за плащане?": "Можете да платите с наложен платеж или с карта.",
    "Мога ли да върна продукт?": "Да, имате право на връщане в рамките на 14 дни."
}
questions = list(faq.keys())
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(questions)

def get_bot_response(user_query):
    query_vec = vectorizer.transform([user_query])
    similarity = cosine_similarity(query_vec, X)
    best_match_idx = similarity.argmax()
    return faq[questions[best_match_idx]] if similarity[0][best_match_idx] > 0.3 else "Свържете се с оператор."

print("Бот:", get_bot_response("Как да платя?"))
```

6. Анализ на настроенията (Sentiment Analysis) в социалните мрежи

Теория: Текстов анализ и обработка на данни. Алгоритми за класификация и концепция за речникови методи.

Практика: Анализ на потребителски мнения и определянето им като позитивни, негативни или неутрални.

Сорс код (Python / TextBlob):

```
from textblob import TextBlob

reviews = [
    "I absolutely love this new smartphone, the camera is amazing!",
    "This is the worst customer service I have ever experienced.",
    "The product is okay, nothing special but works fine."
]

for review in reviews:
    polarity = TextBlob(review).sentiment.polarity
    sentiment = "ПОЗИТИВЕН" if polarity > 0.1 else ("НЕГАТИВЕН" if
polarity < -0.1 else "НЕУТРАЛЕН")
    print(f"Отзив: '{review}' -> Оценка: {sentiment}")
```

7. Система за автоматично резюмиране на статии

Теория: Рекурентни невронни мрежи (RNN, LSTM) и Трансформър архитектури (Transformer, GPT).

Практика: Използване на предварително обучен модел от Hugging Face за резюмиране на дълъг текст.

Сорс код (Python / Transformers):

```
from transformers import pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
text = "Artificial intelligence (AI) is intelligence demonstrated by
machines, as opposed to natural intelligence. AI research focuses on
intelligent agents. The traditional goals include reasoning, knowledge
representation, planning, and learning."

summary = summarizer(text, max_length=40, min_length=10, do_sample=False)
print("Резюме от ИИ:", summary[0]['summary_text'])
```

8. Филтър за откриване на спам и фишинг съобщения

Теория: Принципи на филтрирането на текст. Метрики за оценка на модели (Accuracy, Precision, Recall, F1-score).

Практика: Обучение на класически Наивен Байс класификатор за разпознаване на нежелани писма.

Сорс код (Python / Naive Bayes):

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

emails = ["Спечели безплатна екскурзия сега!", "Здравей, пращам ти
файловете за проекта", "Увеличете доходите си бързо!", "Напомняне за
родителската среща"]
labels = [1, 0, 1, 0]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(emails)
classifier = MultinomialNB().fit(X, labels)

test_vector = vectorizer.transform(["Купи евтини хапчета онлайн
веднага"])
print("Анализ:", "СПАМ" if classifier.predict(test_vector)[0] == 1 else
"ЛЕГИТИМЕН")
```

Категория 3: Анализ на данни и Прогнозиране (Tabular Data & Forecasting)

9. Прогнозиране на цени на недвижими имоти

Теория: Алгоритми за регресия (Линейна регресия, Random Forest). Подготовка и скалиране на данни.

Практика: Линейна регресия за прогноза на цена на база квадратура и стаи.

Сорс код (Python / Linear Regression):

```
import pandas as pd
from sklearn.linear_model import LinearRegression

data = {'Size': [45, 60, 80, 120], 'Rooms': [1, 2, 3, 4], 'Price':
[45000, 65000, 90000, 150000]}
df = pd.DataFrame(data)

model = LinearRegression().fit(df[['Size', 'Rooms']], df['Price'])
predicted_price = model.predict([[80, 3]])
print(f"Прогнозна цена за 80 кв.м. с 3 стаи: {predicted_price[0]:.2f}
EUR")
```

10. Оценка на кредитен риск (Credit Scoring)

Теория: Класификационни алгоритми (Дървета на решенията). Проблемът с небалансираните данни.

Практика: Използване на Decision Tree за автоматично одобряване/отказване на кредити.

Сорс код (Python / Decision Tree):

```
from sklearn.tree import DecisionTreeClassifier

X = [[1200, 22, 0], [4500, 35, 0], [800, 40, 1], [3100, 29, 0]]
y = [0, 1, 0, 1]

clf = DecisionTreeClassifier().fit(X, y)
decision = clf.predict([[3000, 25, 0]])
print("Решение за кредита:", "ОДОБРЕН" if decision[0] == 1 else
"ОТКАЗАН")
```

11. Система за препоръки (Recommendation System) за филми

Теория: Методи за филтриране: съвместно филтриране (Collaborative) и по съдържание (Content-based).

Практика: Косинусово сходство между матрици с потребителски оценки за филми.

Сорс код (Python / Cosine Similarity):

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

ratings = {'Потребител_А': [5, 1, 5, 2], 'Потребител_Б': [1, 5, 2, 5],
'Нов_Потребител': [4, 1, 4, 0]}
df = pd.DataFrame(ratings).T
similarity = cosine_similarity(df)
print("Матрица на сходство:
", similarity)
```

12. Предсказване на текучество на клиенти (Customer Churn Prediction)

Теория: Значение на анализа на потребителското поведение. Feature Engineering от сурови данни.

Практика: Обучение на Random Forest класификатор за определяне на риск от напускане на клиенти.

Сорс код (Python / Random Forest):

```
from sklearn.ensemble import RandomForestClassifier

X = [[1, 5, 50], [24, 0, 20], [3, 6, 55], [48, 1, 15]]
y = [1, 0, 1, 0]

clf = RandomForestClassifier(n_estimators=10).fit(X, y)
will_leave = clf.predict([[2, 6, 55]])
print("Прогноза:", "РИСК ОТ ОТКАЗВАНЕ" if will_leave[0] == 1 else
"ЛОЯЛЕН")
```

Категория 4: Обучение с подсилване (Reinforcement Learning) и Игри

13. Обучение на ИИ агент за игра (Minimax за Морски шах)

Теория: Теория на игрите в ИИ. Алгоритъм Minimax и Алфа-Бета окастряне.

Практика: Опростена симулация на изчисление на най-добър ход.

Сорс код (Python / Концептуален):

```
def minimax(board, depth, is_max):
    # Концептуална рамка за избор на оптимален ход чрез обхождане на
дърво
    print("Алгоритъмът изчислява ходовете напред...")
    return 0

minimax(None, 0, True)
```

14. Автономно преминаване през лабиринт чрез Q-learning

Теория: Концепции на Reinforcement Learning: Агент, Среда, Награда (Reward), Наказание и Q-таблици.

Практика: Обновяване на Q-стойности по формулата на Белман при среща с препятствие/награда.

Сорс код (Python / Q-Learning):

```
import numpy as np

q_table = np.zeros((4, 2)) # 4 състояния, 2 действия
lr, discount = 0.1, 0.9

current_state, action, next_state, reward = 1, 1, 2, -10
q_table[current_state, action] += lr * (reward + discount *
np.max(q_table[next_state]) - q_table[current_state, action])
print("Q-таблица след сблъсък с капан:
", q_table)
```

Категория 5: Киберсигурност и Интернет на нещата (IoT)

15. Откриване на аномалии и мрежови атаки (DDoS)

Теория: Системи за засичане на проникванията (IDS). Неконтролирано обучение и клъстеризация (K-Means).

Практика: Групиране на трафика и маркиране на аномални пакети с висок брой заявки.

Сорс код (Python / K-Means):

```
from sklearn.cluster import KMeans
import numpy as np

traffic = np.array([[5, 10], [8, 12], [6, 8], [500, 1500]])
kmeans = KMeans(n_clusters=2, n_init=10).fit(traffic)
new_traffic = [[450, 1200]]
print("Клъстер за новия трафик:", kmeans.predict(new_traffic),
"(Маркирано като Атака)")
```

16. Смарт управление на енергията в "Умен дом"

Теория: Времеви редове (Time Series Forecasting). Прогнозиране на консумация въз основа на история.

Практика: Скрипт за автоматично засичане на пикови часове на потребление за оптимизация на уреди.

Сорс код (Python / Data Analysis):

```
import numpy as np

hourly_usage = np.array([0.2, 0.2, 0.1, 0.8, 1.5, 2.2, 1.2, 3.4, 4.2,
1.2, 0.5, 0.3])
peak_hours = np.where(hourly_usage > 2.5)[0]
print("Пикови часове засечени от ИИ:", [f"{h}:00" for h in peak_hours])
```

Категория 6: Генеративен ИИ (Generative AI)

17. Генератор на стилизирани изображения (Neural Style Transfer)

Теория: Генеративно-състезателни мрежи (GANs) и Neural Style Transfer. Функции за загуба и Матрица на Грам.

Практика: Извличане на стилови характеристики чрез дефиниране на Матрица на Грам в TensorFlow.

Сорс код (Python / Gram Matrix):

```
import tensorflow as tf
import numpy as np

def gram_matrix(input_tensor):
    result = tf.linalg.einsum('bijc,bijd->bcd', input_tensor,
input_tensor)
    input_shape = tf.shape(input_tensor)
    num_locations = tf.cast(input_shape[1]*input_shape[2], tf.float32)
    return result / num_locations

fake_features = tf.random.normal([1, 10, 10, 64])
print("Форма на Матрицата на Грам за стил:",
gram_matrix(fake_features).shape)
```