ИИ.

Что такое ИИ?

Искусственный интеллект (ИИ) — это область компьютерных наук, которая занимается созданием систем, способных выполнять задачи, требующие человеческого интеллекта. Эти задачи включают в себя обучение, рассуждение, решение проблем, восприятие, понимание естественного языка и даже способность выполнять творческие задачи.

Типы ИИ:

- **Узкий ИИ (ANI)**: Специализируется на одной задаче или наборе задач. Примеры: системы распознавания речи, игра в шахматы.
- **Общий ИИ (AGI)**: Способен выполнять любые интеллектуальные задачи, которые может выполнить человек.
- **Суперинтеллект (ASI)**: Уровень интеллекта, превосходящий человеческий во всех аспектах.

Концепции ИИ.

Машинное Обучение (МО): Подраздел ИИ, где алгоритмы обучаются на данных и улучшают свои функции на основе опыта.

- **Обучение с учителем (Supervised Learning):** Алгоритмы обучаются на размеченных данных.
- **Обучение без учителя (Unsupervised Learning):** Алгоритмы обучаются на неразмеченных данных.
- **Обучение с подкреплением (Reinforcement Learning):** Алгоритмы обучаются путем проб и ошибок с использованием системы вознаграждений и наказаний.

Глубокое Обучение (Deep Learning): Подмножество машинного обучения, использующее многослойные нейронные сети для анализа данных.
Обработка Естественного Языка (NLP): Технологии, позволяющие компьютерам понимать, интерпретировать и генерировать человеческий язык.

Принципы работы ИИ.

- **Сбор Данных:** Данные являются основой для обучения ИИ-моделей. Данные могут быть структурированными (таблицы, базы данных) и неструктурированными (текст, изображения).
- **Предобработка Данных:** Очищение данных, нормализация, выделение признаков.
- **Моделирование:** Выбор и обучение модели на основе подготовленных данных.
- **Оценка:** Проверка точности модели с использованием тестовых данных.
- **Развертывание:** Внедрение модели в реальную среду для использования.
- **Мониторинг и Обновление:** Постоянный мониторинг работы модели и её обновление по мере необходимости.

Библиотеки.

TensorFlow

- **Описание:** TensorFlow — это открытая библиотека для машинного обучения, разработанная Google. Она используется для разработки и обучения моделей машинного обучения и глубокого обучения.
- **Основные возможности:** Поддержка нейронных сетей, обучение и предсказание, визуализация.
- pip install tensorflow

PyTorch

- **Описание:** PyTorch — это библиотека глубокого обучения, разработанная Facebook. Она предоставляет динамическое вычисление графов, что делает её популярной для исследований и разработки прототипов.
- **Основные возможности:** Автоматическое дифференцирование, поддержка CUDA, гибкость в создании моделей.

Keras

- **Описание:** Keras — это высокоуровневый API для нейронных сетей, который работает поверх TensorFlow и других библиотек. Она предназначена для быстрого создания и обучения моделей глубокого обучения.
- **Основные возможности:** Простота использования, модульность, поддержка многослойных сетей.
- pip install keras

Scikit-learn

- **Описание:** Scikit-learn — это библиотека для машинного обучения на Python, которая предоставляет простые и эффективные инструменты для анализа данных и моделирования.
- **Основные возможности:** Регрессия, классификация, кластеризация, предобработка данных.
- pip install scikit-learn

Scikit-Learn

Scikit-learn — это популярная библиотека машинного обучения на Python, которая предоставляет простой и эффективный инструментарий для анализа данных и моделирования. Она включает в себя множество алгоритмов для задач классификации, регрессии, кластеризации и снижения размерности.

Основные Компоненты Scikit-learn

1. **Модели (Estimators):**
   - Модели в Scikit-learn называются "estimators". Это объекты, которые могут обучаться на данных для выполнения задач машинного обучения.
   - Примеры моделей: LinearRegression, RandomForestClassifier, KMeans.
2. **Датасеты:**
   - Scikit-learn предоставляет несколько встроенных датасетов, таких как Iris, Boston Housing, MNIST и другие.
   - Модули: sklearn.datasets.
3. **Методы оценки (Metrics):**
   - Метрики используются для оценки качества моделей, такие как точность, F1-меры, среднеквадратичная ошибка.
   - Модули: sklearn.metrics.

Scikit-Learn

к компонентам также:

Предобработка данных:

- Scikit-learn предоставляет инструменты для масштабирования, нормализации и трансформации данных.
- Модули: sklearn.preprocessing.

Модели выбора (Model Selection):

- Инструменты для разделения данных на тренировочные и тестовые, кросс-валидация, подбор гиперпараметров.
- Модули: sklearn.model_selection.

Scikit-Learn. Основные определения.

- **Датасет (Dataset) -** Набор данных для обучения и оценки моделей.
- **Признаки (Features) -** Индивидуальные характеристики каждого образца в данных.
- **Целевая Метка (Target Label) -** Значение, которое модель предсказывает на основе признаков.
- **Модель (Model) -** Алгоритм, обученный для выполнения задач машинного обучения.
- **Обучение (Training) -** Процесс подбора параметров модели с использованием данных.
- **Тренировочные Данные (Training Data) -** Данные, используемые для обучения модели.
- **Тестовые Данные (Test Data) -** Данные, используемые для оценки производительности модели.
- **Валидаторный Набор (Validation Set) -** Данные для настройки гиперпараметров и предотвращения переобучения.
- **Гиперпараметры (Hyperparameters) -** Параметры модели, настраиваемые до обучения.
- **Обучение failed:**
    - **Переобучение (Overfitting) -** Модель хорошо подстраивается под тренировочные данные, но плохо обобщает.
    - **Недообучение (Underfitting) -** Модель недостаточно хорошо улавливает закономерности в данных.

Scikit-Learn. Основные шаги;

Загрузка и Предобработка Данных:

- Данные могут быть загружены из встроенных датасетов или из внешних источников (файлы CSV, базы данных).
- Предобработка данных включает в себя очистку, масштабирование, кодирование категориальных переменных и другие операции.

Разделение Данных:

- Разделение данных на тренировочный и тестовый наборы для обучения и оценки модели.
- Используется функция `train_test_split` из `sklearn.model_selection`.

Выбор и Обучение Модели:

- Выбор подходящей модели для задачи (например, линейная регрессия для регрессии, SVM для классификации).
- Обучение модели на тренировочных данных с помощью метода `fit`.

Оценка Модели:

- Использование тестовых данных для оценки производительности модели.
- Применение метрик из `sklearn.metrics` для измерения точности, F1-меры и других показателей.

Сохранение и Загрузка Модели:

- Сохранение обученной модели на диск для дальнейшего использования с помощью модуля `joblib`.
- Загрузка модели из файла и использование для предсказаний.

Scikit-Learn

Загрузка данных:

```
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target
```

Разделение данных:

```
from sklearn.model_selection import train_test_split

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

Scikit-Learn

Обучение модели:

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

Оценка модели:

```
from sklearn.metrics import accuracy_score

y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Точность: {accuracy:.2f}')
```

Scikit-Learn

Сохранение/Загрузка модели:

```
import joblib
# Сохранение модели
joblib.dump(clf, 'random_forest_model.pkl')
# Загрузка модели
clf_loaded = joblib.load('random_forest_model.pkl')
```

Предобработка данных:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Общий пример работы с ИИ.

Цель/задание: Разработать модель машинного обучения для распознавания чисел в строковом формате. Модель должна определять, является ли данное значение числом или строкой.

- Примеры чисел: "123", "45.67", "-8".
- Примеры строк: "abc", "12a3", "hello".

создание датасета;

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import re

# Генерация данных
data = ["123", "456", "789", "1.23", "45.67", "-890", "abc", "def", "123abc", "456def", "hello", "world"]
labels = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

df = pd.DataFrame({'data': data, 'label': labels})

# Разделение на тренировочный и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(df['data'], df['label'], test_size=0.2, random_state=42)
# Функция для преобразования строки в признаки
def feature_extraction(s):
   return [
       int(s.isdigit()),
       int(bool(re.match(r'^-?\d+(\.\d+)?$', s)))
   ]

# Преобразование данных в признаки
X_train_features = np.array([feature_extraction(x) for x in X_train])
X_test_features = np.array([feature_extraction(x) for x in X_test])
```

обучение модели;

```
from sklearn.ensemble import RandomForestClassifier


# Обучение модели
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_features, y_train)
```

оценка модели;

```
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Оценка модели
y_pred = clf.predict(X_test_features)
accuracy = accuracy_score(y_test, y_pred)
print(f'Точность: {accuracy:.2f}')

# Матрица ошибок
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

визуализация;

```
from sklearn.model_selection import learning_curve

# Кривая обучения
train_sizes, train_scores, test_scores = learning_curve(clf, X_train_features, y_train, cv=5,
scoring='accuracy', n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 10))

train_scores_mean = np.mean(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)

plt.plot(train_sizes, train_scores_mean, label='Train Score')
plt.plot(train_sizes, test_scores_mean, label='Test Score')
plt.xlabel('Training Size')
plt.ylabel('Accuracy Score')
plt.title('Learning Curve')
plt.legend()
plt.show()
```

полный код с тестами тут:

https://disk.yandex.ru/d/uXtDQQ97FPAwNg