|  |  |
| --- | --- |
| 1. Rekurrent neyron tarmoqlari (RNN) qanday ishlaydi va ularning asosiy farqlari nimada? | Rekurrent neyron tarmoqlari (RNN) sun'iy intellektda vaqtga bog'liq ma'lumotlarni qayta ishlash uchun mo'ljallangan neyron tarmoqlardir. Ular asosan ketma-ketliklar, masalan, matn, ovoz yoki vaqt seriyalari kabi ma'lumotlar bilan ishlashda qo'llaniladi. RNN'larning asosiy xususiyati shundaki, ular o'tgan vaqt bosqichlaridagi ma'lumotlarni eslab qolish qobiliyatiga ega. Bu esa ularni an'anaviy neyron tarmoqlardan farqlaydi.  ▎RNN ning ishlash prinsipi:  1. Kirish va chiqish: RNN kiruvchi ma'lumotni (masalan, biror so'z yoki raqam) qabul qiladi va uni ichki holatga (hidden state) o'tkazadi. Har bir vaqt bosqichida RNN o'zining ichki holatini yangilaydi.  2. Ichki holat: Har bir vaqt bosqichida RNN o'tgan vaqt bosqichidan olingan ichki holatni (hidden state) yangilaydi. Bu ichki holat, avvalgi kirishlar haqida ma'lumotni saqlaydi va yangi kirish bilan birgalikda qayta ishlanadi.  3. Chiqish: Yangilangan ichki holat yordamida RNN chiqish hosil qiladi. Bu chiqish kelajakdagi vaqt bosqichlari uchun prognoz qilishda yoki ma'lumotlarni tasniflashda ishlatilishi mumkin.  ▎RNN ning asosiy farqlari:  1. Vaqtga bog'liqlik: RNN'lar vaqtga bog'liq ma'lumotlarni qayta ishlashda juda samarali, chunki ular avvalgi holatlarni eslab qolish qobiliyatiga ega.  2. O'tgan vaqt bosqichlari bilan bog'lanish: RNN har bir vaqt bosqichida o'z ichki holatini yangilab boradi, bu esa uni ketma-ketlikdagi o'zgarishlarga moslashishga imkon beradi.  3. Qiyinchiliklar: RNN'larda uzun ketma-ketliklar bilan ishlaganda "vanishing gradient" (yo'qolayotgan gradient) muammosi yuzaga kelishi mumkin, bu esa o'qitishni qiyinlashtiradi. Buning uchun LSTM (Long Short-Term Memory) yoki GRU (Gated Recurrent Unit) kabi takomillashtirilgan arxitekturalar ishlab chiqilgan.  4. Qo'llanishi: RNN'lar tabiiy tilni qayta ishlash, ovoz tanish, vaqt seriyalarini tahlil qilish va boshqa ko'plab sohalarda qo'llaniladi.  Umuman olganda, RNN'lar vaqtga bog'liq ma'lumotlarni o'rganish va prognoz qilishda juda foydali vositadir, ammo ularning samaradorligini oshirish uchun ko'pincha LSTM yoki GRU kabi takomillashtirilgan modellardan foydalaniladi. |
| 2. LSTM (Long Short-Term Memory) modeli qanday ishlaydi va uni oddiy RNN’dan farqi nimada? | LSTM (Long Short-Term Memory) modeli – bu rekurrent neyron tarmoqlar (RNN) turidir, u vaqtga bog'liq ma'lumotlarni qayta ishlashda an'anaviy RNN'lar bilan bog'liq muammolarni hal qilish uchun ishlab chiqilgan. LSTM'lar "uzun muddatli" va "qisqa muddatli" xotirani samarali boshqarish imkonini beradi, bu esa ularni uzun ketma-ketliklar bilan ishlashda yanada samarali qiladi.  ▎LSTM ning asosiy tarkibiy qismlari:  1. Kirish (Input) Katlam: Yangi kirish ma'lumotlarini qabul qiladi va ichki holatga ta'sir ko'rsatadi.  2. Unutish (Forget) Katlam: Bu qism ichki holatdan qaysi ma'lumotlarni unutish kerakligini belgilaydi. U 0 dan 1 gacha bo'lgan qiymatlarni oladi, bu esa ma'lum bir ma'lumotni saqlash yoki yo'qotish darajasini ko'rsatadi.  3. Yangilash (Input) Katlam: Bu qism yangi ma'lumotlarni ichki holatga qo'shish uchun qaysi ma'lumotlarni yangilash kerakligini aniqlaydi.  4. Chiqish (Output) Katlam: Yangilangan ichki holatdan chiqishni hosil qiladi.  ▎LSTM ning ishlash prinsipi:  1. Unutish qarori: Unutish katlami avvalgi ichki holatdan qaysi ma'lumotlarni saqlash yoki yo'qotish kerakligini aniqlaydi.  2. Yangilash: Yangilash katlami yangi kirish ma'lumotlarini qo'shadi va ichki holatni yangilaydi.  3. Chiqish: Chiqish katlami yangilangan ichki holatdan chiqishni hosil qiladi.  ▎Oddiy RNN va LSTM o'rtasidagi farqlar:  1. Xotira boshqaruvi: Oddiy RNN'larda ichki holat faqat avvalgi vaqt bosqichidan kelayotgan ma'lumotlar bilan yangilanadi, bu esa uzun muddatli xotirani saqlashda muammolar keltirib chiqaradi. LSTM esa unutish va yangilash katlamlari orqali xotirani yanada samarali boshqaradi.  2. Vanishing gradient muammosi: Oddiy RNN'lar uzun ketma-ketliklar bilan ishlaganda "vanishing gradient" muammosiga duch keladi, bu esa o'qitishni qiyinlashtiradi. LSTM esa bu muammoni hal qiladi, chunki u gradientni uzluksiz saqlashi mumkin.  3. Arxitektura murakkabligi: LSTM modeli oddiy RNN ga qaraganda murakkabroq arxitekturaga ega, chunki u bir nechta katlamlardan iborat. Bu esa LSTM ni ko'p hollarda samaraliroq qiladi, lekin o'qitish jarayonini ham murakkablashtiradi.  4. Qo'llanilishi: LSTM'lar tabiiy tilni qayta ishlash, ovoz tanish, vaqt seriyalarini tahlil qilish va boshqa ko'plab sohalarda keng qo'llaniladi, ayniqsa uzun ketma-ketliklar bilan ishlashda.  Umuman olganda, LSTM modeli RNN'larning cheklovlarini bartaraf etishda va vaqtga bog'liq ma'lumotlarni yanada samarali qayta ishlashda muhim rol o'ynaydi. |
| 3. LSTM modelida qaysi hollarda overfitting yuzaga kelishi mumkin va buni oldini olish usullari qanday? | LSTM (Long Short-Term Memory) modellari murakkab arxitekturaga ega bo'lgani uchun, ularning o'qitilishi jarayonida overfitting (ortiqcha moslashuv) muammosi yuzaga kelishi mumkin. Overfitting, model o'qitish ma'lumotlariga juda yaxshi mos kelganda, lekin yangi, ko'rilmagan ma'lumotlarga yomon natijalar ko'rsatganda sodir bo'ladi. LSTM modellarida overfitting yuzaga kelishi mumkin bo'lgan ba'zi hollarga quyidagilar kiradi:  ▎Overfitting sabablar:  1. Murakkablik: Modelning ortiqcha qatlamlari yoki neyronlari bo'lsa, bu modelning o'qitish ma'lumotlariga juda mos kelishiga olib kelishi mumkin.  2. Kichik o'qitish to'plami: Agar o'qitish to'plami kichik bo'lsa, model ma'lumotlardagi shovqin va tasodifiylikni o'rganib olishi mumkin, bu esa generalizatsiya qobiliyatini pasaytiradi.  3. Ma'lumotlarning noaniqligi: Agar ma'lumotlar juda shovqinli yoki noaniq bo'lsa, model bu shovqinni o'rganib olishi mumkin.  4. O'qitish jarayonining davomiyligi: O'qitishni juda uzoq davom ettirish modelning o'qitish to'pligiga ortiqcha moslashuviga olib kelishi mumkin.  ▎Overfitting ni oldini olish usullari:  1. Dropout: Neyron tarmog'idagi qatlamlarga dropout qo'shish, ya'ni tasodifiy ravishda neyronlarni o'chirib qo'yish orqali modelning generalizatsiya qobiliyatini oshirish mumkin. Bu usul modelning o'zgaruvchanligini oshiradi va overfittingni kamaytiradi.  2. Erta to'xtatish (Early Stopping): O'qitish jarayonida modelning validatsiya to'plamidagi natijalarini kuzatish va natijalar yomonlashganda o'qitishni to'xtatish. Bu modelning ortiqcha moslashuvini oldini olishga yordam beradi.  3. Ma'lumotlarni kengaytirish (Data Augmentation): O'qitish to'plamidagi ma'lumotlarni kengaytirish (masalan, vaqt seriyalarida shovqin qo'shish, ma'lumotlarni burish) orqali modelni yanada mustahkamlash va overfittingni kamaytirish mumkin.  4. Regulyarizatsiya: L2 regulyarizatsiyasi (Tikhonov regulyarizatsiyasi) kabi usullarni qo'llash orqali modelning murakkabligini kamaytirish va overfittingni oldini olish mumkin.  5. Model arxitekturasini soddalashtirish: Modelning qatlamlari yoki neyronlari sonini kamaytirish orqali uning murakkabligini pasaytirish.  6. Katta o'qitish to'plami: Katta va xilma-xil o'qitish to'plamidan foydalanish orqali modelni yanada yaxshi umumlashtirishga yordam berish.  7. Cross-validation: Modelni bir nechta kichik to'plamlarda sinab ko'rish va natijalarni taqqoslash orqali overfittingni aniqlash va oldini olish.  Bu usullar LSTM modellarida overfitting muammosini kamaytirishga yordam beradi va ularni yangi ma'lumotlarga moslashuvchanligini oshiradi. |
| 4. Rekurrent neyron tarmoqlarida vaqtli ketma-ketliklarda ma'lumotni saqlash qanday amalga oshiriladi? | Rekurrent neyron tarmoqlarida (RNN) vaqtli ketma-ketliklarda ma'lumotni saqlash va qayta ishlash jarayoni bir nechta asosiy mexanizmlar orqali amalga oshiriladi. RNN'lar, o'zlarining arxitekturasi tufayli, vaqtli ketma-ketliklardagi ma'lumotlarni saqlab qolish va ularni ishlatish imkoniyatiga ega. Quyida bu jarayonning asosiy jihatlari keltirilgan:  ▎1. Ichki holat (Hidden State):  RNN'lar har bir vaqt bosqichida ichki holat (hidden state) deb ataladigan bir vektorni saqlaydi. Bu ichki holat, oldingi vaqt bosqichidagi ma'lumotlarni va joriy kirish ma'lumotlarini birlashtirib, vaqtli ketma-ketlikdagi kontekstni ifodalaydi.  ▎2. Rekurrent bog'lanish:  RNN'lar o'z ichki holatini yangilash uchun rekurrent bog'lanishni ishlatadi. Har bir vaqt bosqichida, model oldingi vaqt bosqichidagi ichki holatni yangi kirish ma'lumoti bilan birga oladi va yangi ichki holatni hisoblaydi. Bu quyidagi formulada ifodalanadi:  hₜ = f(Wₕ ⋅ hₜ₋₁ + Wₓ ⋅ xₜ + b)  Bu yerda:  • hₜ — joriy vaqt bosqichidagi ichki holat,  • hₜ₋₁ — oldingi vaqt bosqichidagi ichki holat,  • xₜ — joriy kirish ma'lumoti,  • Wₕ va Wₓ — og'irliklar matnlari,  • b — bias vektori,  • f — aktivatsiya funksiyasi (masalan, tanh yoki ReLU).  ▎3. Vaqtli bog'lanish:  RNN'lar vaqtli bog'lanishni saqlash uchun ketma-ketlikdagi har bir elementni navbatma-navbat ishlaydi. Har bir element ishlanayotganda, ichki holat yangilanadi va bu holat keyingi vaqt bosqichida foydalaniladi.  ▎4. Uzun muddatli va qisqa muddatli xotira:  RNN'lar qisqa muddatli xotira sifatida yaxshi ishlaydi, lekin uzoq muddatli bog'lanishlarni saqlashda muammolar yuzaga kelishi mumkin (vanishing gradient muammosi). Buning uchun LSTM (Long Short-Term Memory) va GRU (Gated Recurrent Unit) kabi takomillashgan rekurrent arxitekturalar ishlab chiqilgan.  ▎5. LSTM va GRU:  • LSTM modellarida uchta eshik (input gate, forget gate, output gate) mavjud bo'lib, bu eshiklar ichki holatni boshqarishga yordam beradi va uzoq muddatli bog'lanishlarni saqlash imkonini beradi.  • GRU esa LSTM ga o'xshash, lekin soddaroq arxitekturaga ega bo'lib, ikki eshik (update gate va reset gate) orqali ishlaydi.  ▎6. O'qitish jarayoni:  RNN'lar o'qitish jarayonida backpropagation through time (BPTT) algoritmi yordamida o'qitiladi. Bu usul orqali modelning og'irliklari ichki holatlar orqali tarqatilib, o'qitish ma'lumotlariga moslashadi.  Umuman olganda, rekurrent neyron tarmoqlari vaqtli ketma-ketliklardagi ma'lumotlarni saqlash va qayta ishlashda samarali bo'lishi uchun ichki holatlarni yangilash va rekurrent bog'lanishlardan foydalanadi. LSTM va GRU kabi takomillashgan modellari esa uzoq muddatli bog'lanishlarni saqlashda yanada samaraliroqdir. |
| 5. Ma'lumotlar bazasi bilan LSTM tarmoqlari qanday ishlaydi va ulardan yuqori aniqlikda natija olish uchun qanday optimallashtirishlar kerak? | LSTM (Long Short-Term Memory) tarmoqlari ma'lumotlar bazalari bilan birgalikda ishlashda ko'plab foydali imkoniyatlar taqdim etadi. LSTM'lar vaqtli ketma-ketliklar bilan ishlashda juda samarali bo'lib, ular ma'lumotlarni saqlash va uzoq muddatli bog'lanishlarni eslab qolish qobiliyatiga ega. Ma'lumotlar bazasi bilan LSTM tarmoqlarining integratsiyasi va yuqori aniqlikda natija olish uchun optimallashtirish jarayonlari quyidagi asosiy bosqichlardan iborat:  ▎1. Ma'lumotlarni tayyorlash:  • Ma'lumotlarni tozalash: Ma'lumotlar bazasidan olingan ma'lumotlar toza va to'g'ri bo'lishi kerak. Noto'g'ri yoki yo'q ma'lumotlar modelning aniqligini pasaytirishi mumkin.  • Normallashtirish: Ma'lumotlar normalizatsiya qilinishi kerak (masalan, [0, 1] oralig'ida), bu LSTM tarmog'ining o'qitish jarayonini tezlashtiradi va barqaror qiladi.  • Ketma-ketlikni yaratish: Ma'lumotlar bazasidagi vaqtli ketma-ketliklarni yaratish va ularni LSTM modeliga mos formatga keltirish.  ▎2. Model arxitekturasi:  • Neuronal qatlamlar soni: LSTM modelida qatlamlar sonini va har bir qatlamdagi neyronlar sonini mos ravishda tanlash. Ko'p qatlamli arxitekturalar yanada murakkab munosabatlarni o'rganishga yordam beradi.  • Aktivatsiya funktsiyalari: LSTM qatlamlari uchun tanlangan aktivatsiya funktsiyalari (masalan, tanh yoki ReLU) modelning samaradorligini oshirishi mumkin.  ▎3. O'qitish jarayoni:  • O'qitish algoritmlari: Adam, RMSprop kabi ilg'or optimallashtiruvchi algoritmlar yordamida modelning og'irliklarini yangilash.  • O'qitish parametrlarini sozlash: O'qitish tezligi (learning rate), batch size va epoch sonini optimallashtirish. O'qitish tezligini dinamik ravishda o'zgartirish (learning rate scheduling) ham foydali bo'lishi mumkin.  • Regularizatsiya: Overfitting muammosidan qochish uchun dropout, L1/L2 regularizatsiyasini qo'llash.  ▎4. Modelni baholash va optimallashtirish:  • Kross-valyidatsiya: Modelni baholash uchun kross-valyidatsiya usulidan foydalaning. Bu modelning umumiylashtirish qobiliyatini yaxshilaydi.  • Giperparametrlarni optimallashtirish: Giperparametrlarni (masalan, qatlamlar soni, neyronlar soni, o'qitish tezligi) tahlil qilish va optimallashtirish uchun grid search yoki random search kabi usullarni qo'llash.  ▎5. Modelni tushunish va tahlil qilish:  • Modelning natijalarini tahlil qilish: Modelning chiqishlarini baholash va xatolarni aniqlash. Bu jarayonda confusion matrix, ROC curve kabi usullarni qo'llash mumkin.  • Xususiyatlarni muhimligini baholash: Modelning qaysi xususiyatlarga ko'proq e'tibor berayotganini aniqlash va kerak bo'lsa, ma'lumotlarni yanada boyitish.  ▎6. Integratsiya va real vaqtli ishlov berish:  • Ma'lumotlar bazasi bilan integratsiya: LSTM modelini ma'lumotlar bazasi bilan bog'lash, real vaqtli ma'lumotlarni olish va modellarni yangilash jarayonlarini avtomatlashtirish.  • Modelni monitoring qilish: Modelning ishlashini doimiy ravishda monitoring qilish va kerak bo'lganda qayta o'qitish.  Yuqoridagi bosqichlar orqali LSTM tarmoqlarining ma'lumotlar bazasi bilan ishlashi va yuqori aniqlikda natijalar olish imkoniyatlarini optimallashtirish mumkin. Har bir loyiha uchun maxsus talablar bo'lishi mumkin, shuning uchun har bir bosqichda ehtiyotkorlik bilan yondashish muhimdir. |
| 6. Scikit-learn kutubxonasida klassifikatsiya algoritmlaridan birini amalda qo'llab, uning natijalarini baholang. | Scikit-learn kutubxonasida klassifikatsiya algoritmlaridan birini qo'llash va natijalarini baholash uchun quyidagi misolni keltiraman. Ushbu misolda, biz "Iris" ma'lumotlar to'plamidan foydalanamiz va "K-Nearest Neighbors" (KNN) klassifikatsiya algoritmini ishlatamiz. Keyin modelning natijalarini baholaymiz.  ▎1. Kutubxonalarni import qilish  import numpy as np  import pandas as pd  from sklearn.datasets import load\_iris  from sklearn.model\_selection import train\_test\_split  from sklearn.neighbors import KNeighborsClassifier  from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score  ▎2. Ma'lumotlarni yuklash va tayyorlash  # Iris ma'lumotlar to'plamini yuklaymiz  iris = load\_iris()  X = iris.data # X - xususiyatlar  y = iris.target # y - maqsad o'zgaruvchisi  # Ma'lumotlarni o'qitish va test to'plamlariga bo'lish  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  ▎3. Modelni yaratish va o'qitish  # KNN klassifikatorini yaratamiz  knn = KNeighborsClassifier(n\_neighbors=3)  # Modelni o'qitamiz  knn.fit(X\_train, y\_train)  ▎4. Model yordamida bashorat qilish  # Test to'plamidagi ma'lumotlar uchun bashorat qilish  y\_pred = knn.predict(X\_test)  ▎5. Natijalarni baholash  # Natijalarni baholash  print("Confusion Matrix:")  print(confusion\_matrix(y\_test, y\_pred))  print("\nClassification Report:")  print(classification\_report(y\_test, y\_pred))  print("Accuracy:", accuracy\_score(y\_test, y\_pred))  ▎To'liq kod  Quyida yuqoridagi kodlarning barchasi bir joyda keltirilgan:  import numpy as np  import pandas as pd  from sklearn.datasets import load\_iris  from sklearn.model\_selection import train\_test\_split  from sklearn.neighbors import KNeighborsClassifier  from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score  # Iris ma'lumotlar to'plamini yuklaymiz  iris = load\_iris()  X = iris.data # X - xususiyatlar  y = iris.target # y - maqsad o'zgaruvchisi  # Ma'lumotlarni o'qitish va test to'plamlariga bo'lish  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # KNN klassifikatorini yaratamiz  knn = KNeighborsClassifier(n\_neighbors=3)  # Modelni o'qitamiz  knn.fit(X\_train, y\_train)  # Test to'plamidagi ma'lumotlar uchun bashorat qilish  y\_pred = knn.predict(X\_test)  # Natijalarni baholash  print("Confusion Matrix:")  print(confusion\_matrix(y\_test, y\_pred))  print("\nClassification Report:")  print(classification\_report(y\_test, y\_pred))  print("Accuracy:", accuracy\_score(y\_test, y\_pred))  ▎Natijalar  Ushbu kodni bajarish orqali siz KNN klassifikatorining ishlashini ko'rishingiz mumkin. confusion\_matrix modelning to'g'ri va noto'g'ri bashoratlarini ko'rsatadi. classification\_report esa har bir sinf uchun aniqlik (precision), chaqiruv (recall) va F1 ballarini taqdim etadi. accuracy\_score modelning umumiy aniqligini hisoblaydi.  Ushbu misolda KNN klassifikatori Iris ma'lumotlar to'plamida yaxshi natijalarga erishadi. Har qanday savol yoki qo'shimcha yordam kerak bo'lsa, bemalol so'rang! |
| 7. Regressiya va klassifikatsiya o'rtasidagi asosiy farqlar nimada va ularni qaysi hollarda qo'llash kerak? | Regressiya va klassifikatsiya — bu sun'iy intellekt va mashinani o'rganish sohalarida keng qo'llaniladigan asosiy statistik usullar. Ularning o'rtasidagi asosiy farqlar quyidagilardir:  ▎1. Maqsad:  • Regressiya: Bu usul, ma'lum bir o'zgaruvchi (y) ning boshqa o'zgaruvchilar (x1, x2, ..., xn) ga bog'liqligini aniqlash uchun ishlatiladi. Regresiya natijasi doimiy qiymat (son) bo'ladi. Masalan, uy narxini, vaqtni yoki haroratni bashorat qilish.  • Klassifikatsiya: Bu usul, ma'lum bir ob'ektni belgilangan toifalardan biriga (kategoriya) ajratish uchun ishlatiladi. Klassifikatsiya natijasi esa diskret qiymatlar (kategoriya) bo'ladi. Masalan, elektron pochta xabarini spam yoki spam emas deb tasniflash.  ▎2. Natija:  • Regressiya: Natija doimiy (masalan, 25.5 daraja, 150000 dollar).  • Klassifikatsiya: Natija diskret (masalan, "spam", "spam emas").  ▎3. Qo'llanilishi:  • Regressiya:  • Iqtisodiy ko'rsatkichlarni bashorat qilish (masalan, iqtisodiy o'sish).  • Fizik jarayonlarni modellashtirish (masalan, temperatura o'zgarishini prognoz qilish).  • Mijozlar xarid qilish odatlarini tahlil qilish.  • Klassifikatsiya:  • Tibbiyotda kasalliklarni aniqlash (masalan, saraton yoki sog'lom).  • Mijozlarni segmentlarga ajratish (masalan, yuqori daromadli va past daromadli mijozlar).  • Tasvirlarni tanib olish (masalan, ob'ektlarni tasvirda aniqlash).  ▎4. Algoritmlar:  • Regressiya: Lineer regressiya, polinomial regressiya, lasso regressiya va boshqalar.  • Klassifikatsiya: Logistik regressiya, qaror daraxtlari, random forest, support vector machines (SVM) va neyron tarmoqlar.  Umuman olganda, regressiya va klassifikatsiya o'rtasidagi tanlov sizning maqsadingizga va muammo turiga bog'liqdir. Agar siz doimiy qiymatni bashorat qilmoqchi bo'lsangiz, regressiyani tanlang; agar siz ob'ektlarni toifalarga ajratmoqchi bo'lsangiz, klassifikatsiyani tanlang. |
| 8. Scikit-learn kutubxonasida klasterlash (Clustering) algoritmlarini qo'llash bo'yicha amaliyot bajarib ko'rsating. | Scikit-learn kutubxonasi Python dasturlash tilida klasterlash (clustering) algoritmlarini qo'llash uchun juda qulay vositadir. Quyida sizga KMeans algoritmini ishlatib, klasterlash amaliyotini qanday bajarishni ko'rsataman. Ushbu amaliyotda biz make\_blobs funksiyasidan foydalanib, sun'iy ma'lumotlar to'plamini yaratamiz va KMeans algoritmi yordamida klasterlarni aniqlaymiz.  ▎1. Kutubxonalarni import qilish  Avvalo, kerakli kutubxonalarni import qilamiz:  import numpy as np  import matplotlib.pyplot as plt  from sklearn.datasets import make\_blobs  from sklearn.cluster import KMeans  ▎2. Sun'iy ma'lumotlar to'plamini yaratish  make\_blobs funksiyasi yordamida sun'iy ma'lumotlar to'plamini yaratamiz:  # Sun'iy ma'lumotlar to'plamini yaratamiz  X, y\_true = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60, random\_state=0)  ▎3. KMeans algoritmini qo'llash  Endi KMeans algoritmini qo'llaymiz va klasterlarni aniqlaymiz:  # KMeans modelini yaratamiz  kmeans = KMeans(n\_clusters=4)  kmeans.fit(X)  # Klaster markazlarini olish  centers = kmeans.cluster\_centers\_  # Klasterlarni olish  y\_kmeans = kmeans.predict(X)  ▎4. Natijalarni vizualizatsiya qilish  Natijalarni vizualizatsiya qilish uchun matplotlib kutubxonasidan foydalanamiz:  # Natijalarni chizish  plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')  # Klaster markazlarini chizish  plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')  plt.title("KMeans Klasterlash")  plt.xlabel("X o'qi")  plt.ylabel("Y o'qi")  plt.show()  ▎To'liq kod  Mana, yuqoridagi kodlarning to'liq ko'rinishi:  import numpy as np  import matplotlib.pyplot as plt  from sklearn.datasets import make\_blobs  from sklearn.cluster import KMeans  # Sun'iy ma'lumotlar to'plamini yaratamiz  X, y\_true = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60, random\_state=0)  # KMeans modelini yaratamiz  kmeans = KMeans(n\_clusters=4)  kmeans.fit(X)  # Klaster markazlarini olish  centers = kmeans.cluster\_centers\_  # Klasterlarni olish  y\_kmeans = kmeans.predict(X)  # Natijalarni chizish  plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')  # Klaster markazlarini chizish  plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')  plt.title("KMeans Klasterlash")  plt.xlabel("X o'qi")  plt.ylabel("Y o'qi")  plt.show()  ▎Natija  Ushbu kodni ishga tushirganda siz KMeans algoritmi yordamida klasterlangan ma'lumotlar to'plamini va klaster markazlarini ko'rishingiz mumkin. Har bir klaster boshqalaridan farq qiladi va qizil X belgisi klaster markazlarini ifodalaydi.  Agar sizda qo'shimcha savollar yoki boshqa klasterlash algoritmlari haqida bilmoqchi bo'lsangiz, bemalol so'rang! |
| 9. |  |
| 10. Cross-validation nima va uning mashina o'rganish modellarini baholashdagi roli qanday? | Cross-validation (kross-valitatsiya) — bu mashina o'rganish modellarining umumiylashuvini baholash uchun ishlatiladigan usul. U modelni o'qitish va test qilish jarayonida ma'lumotlarni yanada samarali va adolatli tarzda taqsimlash imkonini beradi. Kross-valitatsiyaning asosiy maqsadi modelning yangi, ko'rilmagan ma'lumotlar ustida qanday ishlashini baholashdir.  ▎Kross-valitatsiyaning asosiy turlari:  1. K-Fold Cross-Validation:  • Ma'lumotlar to'plami K qismga bo'linadi (fold). Har bir qism bir marta test to'plami sifatida ishlatiladi, qolgan K-1 qismi esa o'qitish uchun ishlatiladi. Bu jarayon K marta takrorlanadi, va har bir takrorlashda natijalar yig'iladi.  • O'rtacha natijalar hisoblanadi, bu esa modelning umumiylashuvini yaxshiroq baholash imkonini beradi.  2. Leave-One-Out Cross-Validation (LOOCV):  • Bu kross-valitatsiyaning eng ekstremal shakli bo'lib, har bir ma'lumot nuqtasi alohida test to'plami sifatida ishlatiladi, qolgan barcha nuqtalar esa o'qitish uchun ishlatiladi. Agar ma'lumotlar to'plami n nuqtadan iborat bo'lsa, bu jarayon n marta takrorlanadi.  3. Stratified K-Fold Cross-Validation:  • Bu usulda K-fold kross-valitatsiya amalga oshiriladi, lekin har bir foldda ma'lumotlarning sinflariga nisbatan teng taqsimlanishi ta'minlanadi. Bu asosan sinflararo muvozanatni saqlash muhim bo'lgan vaziyatlarda foydalidir.  ▎Kross-valitatsiyaning afzalliklari:  1. Umumiylashuvni baholash: Modelning yangi ma'lumotlarga qanchalik mos kelishini aniqlash imkonini beradi.  2. Overfittingni kamaytirish: Modelni o'qitishda foydalanilgan ma'lumotlar bilan haddan tashqari moslashuvini oldini olishga yordam beradi.  3. Resurslardan samarali foydalanish: Barcha mavjud ma'lumotlarni o'qitish va test qilishda ishlatishga imkon beradi.  ▎Kross-valitatsiyaning kamchiliklari:  1. Hisoblash xarajatlari: Kross-valitatsiya ko'p marta modelni o'qitishni talab qiladi, bu esa katta ma'lumotlar to'plamlari uchun vaqt va resurslarni talab qilishi mumkin.  2. Shartli bog'liqlik: Agar ma'lumotlar to'plami juda kichik bo'lsa, kross-valitatsiya natijalari ishonchsiz bo'lishi mumkin.  ▎Xulosa  Kross-valitatsiya mashina o'rganish modellarini baholashda muhim rol o'ynaydi, chunki u modelning umumiylashuvini aniqroq baholash imkonini beradi va overfitting xavfini kamaytiradi. Ushbu usul yordamida tadqiqotchilar va dasturchilar o'z modellarini yanada samarali va ishonchli qilishlari mumkin. |
| 11. Modellarni baholashda accuracy, precision, recall kabi mezonlarning farqi nimada va qaysi holatda qaysi mezonlardan foydalanish kerak? | Modellarni baholashda accuracy, precision, va recall kabi mezonlar har biri o'ziga xos ma'lumotlarni taqdim etadi va turli vaziyatlarda qo'llanilishi mumkin. Keling, ularning har birini va qaysi holatlarda foydalanish kerakligini ko'rib chiqamiz.  ▎1. Accuracy (Aniqlik)  • Aniqlik — bu to'g'ri tasniflangan namunalar sonining umumiy namunalar soniga nisbati.  • Formula:  Accuracy = To'g'ri tasniflanganlar / Umumiy namunala}  • Qachon foydalanish kerak:  • Accuracy oddiy va tushunarli bo'lgani uchun ko'plab vaziyatlarda foydalidir, lekin u muvozanatsiz sinflar mavjud bo'lganda (ya'ni, bir sinf boshqa sinflardan ancha ko'p bo'lsa) ishonchli emas. Masalan, agar sizning ma'lumotlaringizda 95% "manfiy" va 5% "ijobiy" namunalar bo'lsa, model faqat "manfiy" deb tasniflash orqali 95% aniqlikka erishishi mumkin.  ▎2. Precision (Aniqlik)  • Aniqlik — bu model tomonidan ijobiy deb belgilangan namunalar ichida haqiqatan ham ijobiy bo'lganlarning foizi.  • Formula:  Precision = To'g'ri ijobiylar / To'g'ri ijobiyla + Noto'g'ri ijobiylar}  • Qachon foydalanish kerak:  • Precision yuqori bo'lishi kerak bo'lgan vaziyatlarda foydalidir, masalan, tibbiyotda kasalliklarni aniqlashda. Agar model noto'g'ri ijobiy natija bersa (ya'ni, sog'lom odamni kasal deb belgilasa), bu salbiy oqibatlarga olib kelishi mumkin.  ▎3. Recall (Qaytarish)  • Qaytarish — bu haqiqatan ham ijobiy bo'lgan namunalar ichida model tomonidan to'g'ri aniqlangan ijobiylarning foizi.  • Formula:  Recall = To'g'ri ijobiylar / To'g'ri ijobiyla + Noto'g'ri manfiylar}  • Qachon foydalanish kerak:  • Recall yuqori bo'lishi kerak bo'lgan vaziyatlarda, masalan, jinoyatchilarni aniqlash yoki saraton kasalligini erta bosqichda aniqlashda. Agar model noto'g'ri manfiy natija bersa (ya'ni, kasal odamni sog'lom deb belgilasa), bu juda xavfli bo'lishi mumkin.  ▎Xulosa  • Accuracy — umumiy aniqlikni baholash uchun yaxshi, lekin sinflar muvozanati bo'lmagan holatlarda ishonchli emas.  • Precision — noto'g'ri ijobiylar muhim bo'lgan vaziyatlarda foydali.  • Recall — noto'g'ri manfiylar muhim bo'lgan vaziyatlarda afzal ko'riladi.  Agar sizning maqsadingiz faqat biror mezonni baholash bo'lsa, F1-score (precision va recall o'rtasidagi muvozanat) ham qo'llanilishi mumkin. U quyidagi formula bilan hisoblanadi:  F1 Score = 2 × Precision × Recall / Precisio + Recall}  Har bir mezonning o'ziga xos afzalliklari bor, shuning uchun qaysi mezondan foydalanish kerakligini tanlashda vaziyatning kontekstini hisobga olish muhimdir. |
| 12. Pipelines nima va ular mashina o'rganish jarayonini qanday tartibga soladi? | Pipelines (quvurlar) — bu mashina o'rganish jarayonini tartibga solish va avtomatlashtirish uchun ishlatiladigan tuzilmalar. Ular ma'lumotlarni tayyorlashdan tortib modelni o'qitish va baholashgacha bo'lgan barcha bosqichlarni birlashtiradi. Pipelines yordamida jarayonlarni yanada samarali va takrorlanadigan qilish mumkin.  ▎Pipelines ning asosiy komponentlari  1. Ma'lumotlarni tayyorlash:  • Ma'lumotlarni to'plash, tozalash va transformatsiya qilish jarayonlari.  • Misol: yo'qolgan qiymatlarni to'ldirish, kategorik o'zgaruvchilarni kodlash, normalizatsiya va standartizatsiya.  2. Xususiyatlarni tanlash va muhandislik:  • Modelga eng mos keladigan xususiyatlarni tanlash va yangi xususiyatlar yaratish.  • Misol: xususiyatlarni birlashtirish, polinom xususiyatlar yaratish.  3. Modelni o'qitish:  • Tanlangan algoritmni ishlatib modelni o'qitish.  • Misol: regressiya, tasniflash, klasterlash algoritmlari.  4. Modelni baholash:  • Modelning samaradorligini baholash uchun test ma'lumotlari bilan sinovdan o'tkazish.  • Misol: aniqlik, recall, precision va F1-score kabi metrikalarni hisoblash.  5. Modelni joylashtirish:  • O'qitilgan modelni haqiqiy muhitda ishlatish uchun tayyorlash.  • Misol: REST API orqali xizmat ko'rsatish.  ▎Pipelines ning afzalliklari  1. Tartibga solish:  • Har bir bosqichni aniq belgilash orqali jarayonni tartibga soladi, bu esa murakkablikni kamaytiradi.  2. Takrorlanish:  • Bir xil jarayonni takroran ishlatish imkonini beradi, bu esa eksperimentlarni osonlashtiradi.  3. Avtomatlashtirish:  • Jarayonlarning avtomatlashtirilishi vaqtni tejaydi va inson xatolarini kamaytiradi.  4. Modularlik:  • Har bir bosqich alohida modullar sifatida ko'rib chiqilishi mumkin, bu esa ularni yangilash yoki o'zgartirishni osonlashtiradi.  5. Yana ishlatish:  • Turli xil loyihalarda bir xil pipeline ni qayta ishlatish imkonini beradi, bu vaqtni tejaydi va samaradorlikni oshiradi.  ▎Mashina o'rganishda pipelines dan foydalanish  Mashina o'rganish jarayonida pipelines odatda quyidagi kutubxonalar yordamida yaratiladi:  • scikit-learn: Python dasturlash tilida mashhur kutubxona bo'lib, u pipeline yaratish uchun qulay interfeys taqdim etadi.  • TensorFlow va Keras: chuqur o'rganish uchun kuchli kutubxonalar bo'lib, ular ham pipeline yaratishga yordam beradigan vositalarni taqdim etadi.  • Apache Airflow yoki Kubeflow: katta hajmdagi ma'lumotlar bilan ishlashda pipeline ni boshqarish va avtomatlashtirish uchun ishlatiladi.  ▎Xulosa  Pipelines — bu mashina o'rganish jarayonini tartibga soluvchi va avtomatlashtiruvchi muhim vositalardir. Ular har bir bosqichni aniq belgilab, jarayonni samarali va takrorlanadigan qiladi, bu esa tadqiqotlar va ishlab chiqish jarayonlarini tezlashtiradi. |
| 13. Regression modellarida MSE va R-squared mezonlarini qanday qo'llash mumkin va ularning natijalari nimani anglatadi? | MSE (Mean Squared Error) va R-squared (R²) regressiya modellarini baholashda keng qo'llaniladigan ikki muhim mezondir. Ularning har biri modelning samaradorligini o'lchashda turli jihatlarni yoritadi.  ▎1. MSE (Mean Squared Error)  MSE — bu o'rtacha kvadrat xatolarni o'lchovidir. U quyidagi formula orqali hisoblanadi:  MSE = 1 / n ∑ᵢ₌₁ⁿ (yᵢ - ^yᵢ)²  • yᵢ — haqiqiy qiymatlar.  • ^yᵢ — model tomonidan bashorat qilingan qiymatlar.  • n — ma'lumotlar to'plamidagi namunalar soni.  ▎Natijalar va ularning ma'nosi:  • MSE = 0: Model mukammal, ya'ni barcha bashoratlar haqiqiy qiymatlarga to'g'ri keladi.  • MSE > 0: Xato mavjud, qanchalik katta bo'lsa, modelning bashorati shuncha yomonroq.  • MSE ning kichik qiymati modelning yaxshi ishlashini anglatadi.  Afzalliklari:  • MSE kvadrat xatolarni o'lchaydi, bu esa katta xatolarni ko'proq ta'sir qiladi. Bu, ayniqsa, katta xatolarni kamaytirish kerak bo'lgan hollarda foydalidir.  Kamchiliklari:  • MSE ning o'lchov birligi haqiqiy qiymatlarning kvadratiga teng bo'lgani uchun, natijalarni talqin qilish qiyin bo'lishi mumkin.  ▎2. R-squared (R²)  R-squared — bu modelning ma'lumotlardagi o'zgarishni qanchalik yaxshi tushuntirayotganini ko'rsatadigan ko'rsatkich. U quyidagi formula orqali hisoblanadi:  R² = 1 - SSᵣₑₛ / Sₜₒₜ}  • SSᵣₑₛ — qolgan kvadratlar summasi (model xatolari).  • SSₜₒₜ — umumiy kvadratlar summasi (ma'lumotlarning o'zgarishi).  ▎Natijalar va ularning ma'nosi:  • R² = 1: Model barcha o'zgarishni mukammal tushuntiradi.  • R² = 0: Model hech qanday o'zgarishni tushuntirmaydi, ya'ni bashoratlar faqat o'rtacha qiymatga teng.  • 0 < R² < 1: Model ma'lumotlardagi o'zgarishning bir qismini tushuntiradi. R² ning yuqori qiymati modelning yaxshi ishlashini anglatadi.  Afzalliklari:  • R² intuitiv ko'rsatkich bo'lib, modelning tushuntirish qobiliyatini ifodalaydi.  Kamchiliklari:  • R² har doim oshishi mumkin, agar modelga yangi xususiyatlar qo'shilsa, bu esa ortiqcha moslashuvga olib kelishi mumkin. Shuning uchun, ko'p xususiyatli modellar uchun korreksiyalangan R² (Adjusted R²) ko'proq ma'qul hisoblanadi.  ▎Xulosa  MSE va R-squared regressiya modellarini baholashda muhim rol o'ynaydi. MSE modelning aniqligini o'lchasa, R-squared modelning tushuntirish qobiliyatini ko'rsatadi. Har ikkala mezon ham birgalikda ishlatilganda, modelning samaradorligini yanada aniqroq baholash imkonini beradi. |
| 14. Cross-validation jarayoni qaysi qadamlarni o'z ichiga oladi va uning amaliy qo'llanishini ko'rsating. | Kross-validatsiya jarayoni sun'iy intellekt va mashinani o'qitish sohasida modelni baholash va uning umumiylashuv qobiliyatini oshirish uchun ishlatiladi. Kross-validatsiya jarayoni quyidagi qadamlarni o'z ichiga oladi:  ▎Kross-validatsiya jarayoni  1. Ma'lumotlarni tayyorlash:  • Ma'lumotlar to'plamini tozalash va tayyorlash. Bu bosqichda, yo'qolgan qiymatlar, anomaliyalarning bartaraf etilishi va xususiyatlarning normalizatsiyasi amalga oshiriladi.  2. Ma'lumotlarni bo'lish:  • Ma'lumotlar to'plamini k qismga bo'lish (masalan, 5 yoki 10). Har bir qism (fold) trening va test to'plamlariga ajratiladi.  3. Modelni o'qitish va baholash:  • Har bir fold uchun:  • k-1 qismni trening to'plami sifatida foydalaning.  • Qolgan 1 qismni test to'plami sifatida foydalaning.  • Modelni trening to'plamida o'qiting.  • Test to'plamida modelning samaradorligini baholang (masalan, MSE, R² yoki boshqa ko'rsatkichlar yordamida).  4. Natijalarni yig'ish:  • Har bir folddan olingan baholash ko'rsatkichlarini yig'ing va ularning o'rtachasini hisoblang.  5. Modelni baholash:  • Olingan o'rtacha ko'rsatkichlar yordamida modelning umumiy ishlashini baholang.  6. Modelni optimallashtirish (agar kerak bo'lsa):  • Agar kross-validatsiya natijalari qoniqarli bo'lmasa, model parametrlari yoki xususiyatlarni o'zgartirib, jarayonni takrorlang.  ▎Amaliy qo'llanilishi  ▎Misol: Tasniflash modeli uchun kross-validatsiya  Faraz qilaylik, sizda elektron pochta xabarlarini spam va normal xabarlar sifatida tasniflash uchun ma'lumotlar to'plami mavjud. Ushbu ma'lumotlar to'plamida xabar matni, yuboruvchi manzili va boshqa xususiyatlar mavjud.  1. Ma'lumotlarni tayyorlash:  • Xabar matnini tozalang (masalan, maxsus belgilarni olib tashlash), xususiyatlarni chiqarish va normalizatsiya qilish.  2. Ma'lumotlarni bo'lish:  • Ma'lumotlar to'plamini 10 ta foldga bo'lish.  3. Modelni o'qitish va baholash:  • Har bir fold uchun:  • 9 ta foldni trening to'plami sifatida foydalaning.  • 1 ta foldni test to'plami sifatida foydalaning.  • Masalan, logistic regression yoki random forest modelini o'qiting.  • Test to'plamida modelning aniqligini (accuracy) hisoblang.  4. Natijalarni yig'ish:  • Har bir folddan olingan aniqlik ko'rsatkichlarini yozing va ularning o'rtachasini hisoblang.  5. Modelni baholash:  • Olingan o'rtacha aniqlik orqali modelning umumiy samaradorligini baholang.  6. Modelni optimallashtirish:  • Agar aniqlik past bo'lsa, model arxitekturasini o'zgartiring (masalan, yangi xususiyatlar qo'shing yoki hiperparametrlarni sozlang) va jarayonni takrorlang.  ▎Xulosa  Kross-validatsiya sun'iy intellektda modelning umumiylashuv qobiliyatini baholash va oshirishda muhim rol o'ynaydi. U turli bo'lishlarga asoslangan holda modelning haqiqiy dunyoda qanday ishlashini aniqlashga yordam beradi. Bu jarayon mashinani o'qitish mutaxassislari tomonidan keng qo'llaniladi va samarali natijalarga erishishga yordam beradi. |
| 15. Modelni o'rgatish jarayonida giperparametrlarni optimallashtirish qanday amalga oshiriladi? |  |
| 16. Nomutanosib ma'lumotlar (Imbalance data) nima va ularga duch kelganda qanday muammolar yuzaga keladi? | Nomutanosib ma'lumotlar (imbalance data) — bu tasniflash muammolarida uchraydigan holat bo'lib, bunda ma'lumotlar to'plamidagi sinflar orasida sezilarli farq mavjud. Boshqacha aytganda, bir sinf (masalan, ijobiy sinf) boshqa sinfga (masalan, salbiy sinf) nisbatan ancha ko'p yoki kam bo'lishi mumkin. Sun'iy intellekt va mashinani o'rganish sohalarida bu holat bir qator muammolarni keltirib chiqarishi mumkin.  ▎Muammolar  1. Modelning qiyinligi:  • Nomutanosib ma'lumotlar bilan o'qitilgan model ko'p sonli sinflarga ega bo'lgan to'plamda yaxshi ishlasa-da, kam sonli sinflarni e'tiborsiz qoldirishi mumkin. Bu holatda modelning aniqligi yuqori bo'lishi mumkin, lekin kam sonli sinflar uchun natijalar past bo'ladi.  2. Aniqlikning yuqori ko'rsatkichi:  • Model ko'p sonli sinfga to'g'ri tasniflashni o'rganadi va natijada aniqlik ko'rsatkichi yuqori bo'lishi mumkin. Ammo bu faqat ko'p sonli sinflar uchun to'g'ri bo'ladi va kam sonli sinflar e'tiborsiz qolishi mumkin.  3. Xato turlari:  • Kam sonli sinflar uchun xato turlari (false negatives) ko'payishi mumkin, bu esa muhim holatlarni e'tiborsiz qoldirishga olib keladi. Masalan, kasalliklarni aniqlashda ijobiy natijalarni e'tiborsiz qoldirish xavfli bo'lishi mumkin.  4. Baholash ko'rsatkichlari:  • Oddiy aniqlik ko'rsatkichlari (accuracy) nomutanosib ma'lumotlar uchun yolg'on natijalar berishi mumkin. Buning o'rniga F1 balli, precision va recall kabi boshqa ko'rsatkichlardan foydalanish tavsiya etiladi.  5. O'qitish jarayoni:  • Nomutanosib ma'lumotlar bilan o'qitish jarayoni murakkabroq bo'lishi mumkin, chunki modelni balanslash va kam sinflarni yaxshiroq o'rganishga majbur qilish kerak.  ▎Yechimlar  Nomutanosib ma'lumotlar bilan ishlashda quyidagi usullarni qo'llash mumkin:  1. Ma'lumotlarni muvozanatlash:  • Oversampling: Kam sonli sinf uchun ma'lumotlarni ko'paytirish (masalan, SMOTE usuli).  • Undersampling: Ko'p sonli sinfdan ma'lumotlarni kamaytirish.  2. Maxsus algoritmlar:  • Nomutanosib ma'lumotlarga moslashtirilgan algoritmlardan foydalanish (masalan, cost-sensitive learning).  3. Baholash ko'rsatkichlarini o'zgartirish:  • F1 balli, precision va recall kabi ko'rsatkichlardan foydalanish.  4. Ensemble usullari:  • Bir nechta modelni birlashtirib, ularning natijalarini birlashtirish (masalan, bagging yoki boosting).  5. Anomaliyalarni aniqlash:  • Kam sonli sinflar anomal sifatida qaralishi mumkin va bu jarayonda maxsus metodlardan foydalanish mumkin.  Nomutanosib ma'lumotlar bilan ishlashda ehtiyotkorlik talab etiladi, chunki bu holat modelning haqiqiy samaradorligini sezilarli darajada pasaytirishi mumkin. |
| 17. SMOTE texnikasini qo'llab, nomutanosib ma'lumotlarni qanday balanslash mumkinligini tushuntirib bering. |  |
| 18. Modelni saqlash va yuklash uchun joblib yoki pickle dan foydalanish usullarini amalda ko'rsating. | Modelni saqlash va yuklash uchun joblib yoki pickle kutubxonalaridan foydalanish juda qulay. Ushbu kutubxonalar Python ob'ektlarini (masalan, mashinani o'rganish modellari) faylga saqlash va keyinchalik ularni qayta yuklash imkonini beradi. Quyida ikkala usulni amalda ko'rsataman.  ▎1. Joblib yordamida modelni saqlash va yuklash  joblib — bu ayniqsa katta numpy massivlarini saqlash uchun optimallashtirilgan kutubxona. Mashinani o'rganish modellarini saqlash va yuklash uchun juda foydalidir.  ▎O'rnatish  Agar sizda joblib o'rnatilmagan bo'lsa, quyidagi buyruq bilan o'rnating:  pip install joblib  ▎Misol:  from sklearn.datasets import load\_iris  from sklearn.ensemble import RandomForestClassifier  from joblib import dump, load  # Ma'lumotlarni yuklash  data = load\_iris()  X, y = data.data, data.target  # Modelni yaratish  model = RandomForestClassifier()  model.fit(X, y)  # Modelni saqlash  dump(model, 'random\_forest\_model.joblib')  # Modelni yuklash  loaded\_model = load('random\_forest\_model.joblib')  # Yuklangan model bilan bashorat qilish  predictions = loaded\_model.predict(X)  print(predictions)  ▎2. Pickle yordamida modelni saqlash va yuklash  pickle — bu Python ob'ektlarini saqlash va qayta yuklash uchun standart kutubxona.  ▎Misol:  import pickle  from sklearn.datasets import load\_iris  from sklearn.ensemble import RandomForestClassifier  # Ma'lumotlarni yuklash  data = load\_iris()  X, y = data.data, data.target  # Modelni yaratish  model = RandomForestClassifier()  model.fit(X, y)  # Modelni saqlash  with open('random\_forest\_model.pkl', 'wb') as file:  pickle.dump(model, file)  # Modelni yuklash  with open('random\_forest\_model.pkl', 'rb') as file:  loaded\_model = pickle.load(file)  # Yuklangan model bilan bashorat qilish  predictions = loaded\_model.predict(X)  print(predictions)  ▎Xulosa  Ikkala usul ham modelni saqlash va yuklash uchun samarali bo'lib, ular orasidagi tanlov asosan shaxsiy afzalliklar va ma'lumotlar hajmiga bog'liq. joblib ko'proq katta ma'lumotlar bilan ishlash uchun optimallashtirilgan, shuning uchun katta modellar uchun tavsiya etiladi. pickle esa oddiy va keng tarqalgan standart kutubxona hisoblanadi. |
| 19. Nomutanosib ma'lumotlar bilan ishlaganda qaysi baholash mezonlari eng to'g'ri natijani beradi? | Nomutanosib ma'lumotlar bilan ishlaganda sun'iy intellekt tizimlarining samaradorligini baholash uchun bir nechta mezonlar mavjud. Ushbu mezonlar quyidagilarni o'z ichiga oladi:  1. Aniqlik (Accuracy): Modelning to'g'ri klassifikatsiya qilingan namunalari umumiy namunalarga nisbatan foizini ko'rsatadi.  2. F1-ball (F1 Score): Aniqlik va qaytarish (recall) o'rtasidagi muvozanatni ifodalaydi. Nomutanosib ma'lumotlar bo'lganda, bu mezon muhim ahamiyatga ega, chunki u klassifikatsiyaning to'g'riligi va qamrovini birlashtiradi.  3. Qaytarish (Recall): Modelning haqiqiy ijobiy namunalardan qancha foizini to'g'ri aniqlashini ko'rsatadi. Bu mezon nomutanosib ma'lumotlar bilan ishlaganda juda muhim, chunki ba'zi klasslar boshqa klasslarga nisbatan kamroq namunalarga ega bo'lishi mumkin.  4. Aniqlik (Precision): Model ijobiy deb belgilagan namunalarning nechtasi haqiqatan ham ijobiy ekanligini ko'rsatadi. Bu ham nomutanosib ma'lumotlar uchun muhimdir.  5. ROC-AUC: Receiver Operating Characteristic curve (ROC) va uning ostidagi maydon (AUC) modelning turli thresholdlarda ijobiy va salbiy klasslarni ajratish qobiliyatini baholaydi.  6. Kross-valyatsiya (Cross-validation): Modelni turli ma'lumotlar bo'laklari ustida sinab ko'rish orqali uning umumiylashuv qobiliyatini baholash.  7. Klasslar balansi (Class Balance): Nomutanosib ma'lumotlarda har bir klassning taqsimoti va ularning ta'siri haqida ma'lumot beradi.  Ushbu mezonlarni birgalikda qo'llash, nomutanosib ma'lumotlar bilan ishlaganda modelning samaradorligini aniqroq baholash imkonini beradi. |
| 20. Nomutanosib ma'lumotlar bilan ishlashda modellarning aniqligini oshirish uchun qaysi usullardan foydalanish kerak? | Nomutanosib ma'lumotlar bilan ishlashda modellarning aniqligini oshirish uchun bir qator usullar va strategiyalarni qo'llash mumkin. Quyida ba'zi asosiy usullar keltirilgan:  ▎1. Ma'lumotlarni balanslash  • Oversampling: Kam uchraydigan klasslarning ma'lumotlarini ko'paytirish (masalan, SMOTE — Synthetic Minority Over-sampling Technique).  • Undersampling: Ko'p uchraydigan klasslardan ma'lumotlarni kamaytirish.  • Data Augmentation: Kam uchraydigan klasslar uchun yangi misollar yaratish, masalan, tasvirlarni o'zgartirish yoki shovqin qo'shish.  ▎2. Modelni optimallashtirish  • Hiperparametrlarni tuning: Modelning hiperparametrlarini optimallashtirish orqali aniqlikni oshirish.  • Ensemble Learning: Bir nechta modellarni birlashtirib, yakuniy qarorni qabul qilish (masalan, Bagging, Boosting).  ▎3. Xususiyatlarni tanlash va muhandislik qilish  • Xususiyatlarni tanlash: Eng muhim xususiyatlarni tanlash orqali modelning murakkabligini kamaytirish va aniqlikni oshirish.  • Xususiyatlarni muhandislik qilish: Yangi xususiyatlar yaratish yoki mavjudlarini o'zgartirish orqali ma'lumotlarning sifatini yaxshilash.  ▎4. Ma'lumotlarni to'g'rilash  • Ma'lumotlarni tozalash: Noto'g'ri yoki yo'q ma'lumotlarni olib tashlash.  • Ma'lumotlarni normalizatsiya va standartlashtirish: Ma'lumotlarning tarqalishini yaxshilash uchun normalizatsiya yoki standartlashtirish.  ▎5. Modelni baholash va o'zgartirish  • Kross-validatsiya: Modelning aniqligini baholashda kross-validatsiyani qo'llash, bu modelning umumiylashuvini yaxshilaydi.  • Modelni yangilash: Yangi ma'lumotlar bilan modelni muntazam ravishda yangilab turish.  ▎6. Qaror qabul qilish strategiyalari  • Threshold o'zgartirish: Ijobiy va salbiy klasslar o'rtasidagi qaror qabul qilish thresholdini o'zgartirish orqali aniqlikni oshirish.  • Class Weights qo'llash: Nomutanosiblikni hisobga olish uchun kam uchraydigan klasslarga og'irlik berish.  ▎7. Tahlil va vizualizatsiya  • Confusion Matrix tahlili: Modelning qaysi klasslarda kamchiliklari borligini aniqlash va ularni tuzatish.  • Feature Importance tahlili: Modelda qaysi xususiyatlarning eng muhimligini aniqlash va ularga e'tibor qaratish.  ▎Xulosa  Nomutanosib ma'lumotlar bilan ishlaganda, aniqlikni oshirish uchun bir nechta usullarni birlashtirib qo'llash tavsiya etiladi. Har bir usulning samaradorligi vaziyatga qarab farq qilishi mumkin, shuning uchun eksperimentlar o'tkazish muhimdir. |
| 21. Jirada yangi loyiha yarating va unga asosiy komponentlar (epic, story, task) qo‘shing. | Jira'da yangi loyiha yaratish va unga asosiy komponentlar (epic, story, task) qo'shish jarayoni quyidagi bosqichlardan iborat:  ▎1. Yangi loyiha yaratish  1. Jira'ga kirish: Jira hisobingizga kiring.  2. Loyihalar bo'limiga o'tish: Yuqori menyudan "Projects" (Loyihalar) bo'limini tanlang.  3. Yangi loyiha yaratish: "Create Project" (Loyiha yaratish) tugmasini bosing.  4. Loyihaning turini tanlash: Loyihangiz uchun kerakli shablonni tanlang (masalan, Scrum, Kanban).  5. Loyihaning nomi va kalit so'zini kiritish: Loyihangizga nom bering va kalit so'z (key) tanlang.  6. Loyihani yaratish: "Create" (Yaratish) tugmasini bosing.  ▎2. Epic qo'shish  1. Epiklar bo'limiga o'tish: Loyihangiz ichida "Backlog" (Orqaga qaytish) bo'limiga o'ting.  2. Epiklar panelini ochish: Chap tomondagi panelda "Epics" bo'limini tanlang.  3. Yangi epik yaratish: "Create Epic" (Epik yaratish) tugmasini bosing.  4. Epikning nomi va ta'rifini kiritish: Epikning nomini va qisqacha ta'rifini kiriting.  5. Epikni saqlash: "Create" (Yaratish) tugmasini bosing.  ▎3. Story qo'shish  1. Story qo'shish: "Backlog" bo'limida "Create Issue" (Muammo yaratish) tugmasini bosing.  2. Story turini tanlash: Muammo turini "Story" sifatida tanlang.  3. Storyning nomi va ta'rifini kiritish: Story uchun nom va ta'rifni kiriting.  4. Epikga bog'lash: Agar storyni epikka bog'lamoqchi bo'lsangiz, "Epic Link" maydonida tegishli epikni tanlang.  5. Storyni saqlash: "Create" (Yaratish) tugmasini bosing.  ▎4. Task qo'shish  1. Task qo'shish: "Backlog" bo'limida yana "Create Issue" (Muammo yaratish) tugmasini bosing.  2. Task turini tanlash: Muammo turini "Task" sifatida tanlang.  3. Taskning nomi va ta'rifini kiritish: Task uchun nom va ta'rifni kiriting.  4. Storyga bog'lash: Agar taskni storyga bog'lamoqchi bo'lsangiz, "Story Link" maydonida tegishli storyni tanlang.  5. Taskni saqlash: "Create" (Yaratish) tugmasini bosing.  ▎Xulosa  Bu jarayon orqali siz Jira'da yangi loyiha yaratishingiz va unga asosiy komponentlarni qo'shishingiz mumkin. Har bir komponent loyihangizni boshqarishda muhim rol o'ynaydi, shuning uchun ularni to'g'ri tashkil qilish juda muhimdir. Agar sizga qo'shimcha yordam kerak bo'lsa, bemalol so'rang! |
| 22. Jirada yangi “issue” (muammo) yarating va uni tegishli jamoa a’zolariga tayinlang. Unga qanday tasnif va prioritet berish mumkinligini ko‘rsating | Jira'da yangi "issue" (muammo) yaratish va uni jamoa a'zolariga tayinlash jarayoni quyidagi bosqichlardan iborat:  ▎1. Yangi muammo yaratish  1. Jira'ga kirish: Jira hisobingizga kiring.  2. Loyihangizni tanlang: Yuqori menyudan kerakli loyihani tanlang.  3. Muammo yaratish: Loyihangiz ichida "Create" (Yaratish) tugmasini bosing.  ▎2. Muammo ma'lumotlarini kiritish  1. Muammo turini tanlash: "Issue Type" (Muammo turi) maydonidan kerakli muammo turini tanlang (masalan, Task, Bug, Story).  2. Muammo nomi: "Summary" (Qisqacha) maydoniga muammo nomini kiriting.  3. Ta'rif: "Description" (Ta'rif) maydoniga muammo haqida batafsil ma'lumot kiriting.  ▎3. Muammo tasnifini berish  1. Tasnif: "Labels" (Tasniflar) maydoniga muammo bilan bog'liq teglarni kiriting. Bu sizga muammolarni guruhlashda yordam beradi.  2. Kategoriyalar: Agar loyiha doirasida maxsus tasniflar mavjud bo'lsa, ularni tanlang.  ▎4. Prioritet berish  1. Prioritet: "Priority" (Prioritet) maydonidan muammo uchun mos prioritetni tanlang (masalan, Blocker, Critical, Major, Minor, Trivial).  • Prioritet muammolarni hal qilishda qanchalik tezlikda e'tibor berilishi kerakligini belgilaydi.  ▎5. Jamoa a'zolariga tayinlash  1. Assignee (Tayinlangan): "Assignee" (Tayinlangan) maydonida muammoni hal qiladigan jamoa a'zosini tanlang. Agar kimnidir tayinlamoqchi bo'lsangiz, ularning ismini yozing va ro'yxatdan tanlang.  2. Reporter: Sizning ismingiz avtomatik ravishda "Reporter" (Hisobot beruvchi) maydonida ko'rsatiladi, lekin agar kerak bo'lsa, boshqa biror kishini tayinlashingiz mumkin.  ▎6. Qo'shimcha parametrlar  • Due Date (Muddat): Agar muammo uchun belgilangan muddat bo'lsa, uni kiriting.  •Attachment (Ilova): Agar kerak bo'lsa, muammoga fayllarni qo'shishingiz mumkin.  ▎7. Muammoni saqlash  1. Muammoni yaratish: Barcha ma'lumotlarni kiritgandan so'ng, "Create" (Yaratish) tugmasini bosing.  ▎Xulosa  Endi siz yangi muammo yaratdingiz va uni tegishli jamoa a'zolariga tayinladingiz, shuningdek, muammoga tasnif va prioritet berdingiz. Bu jarayon yordamida siz jamoangizning ishini yanada samarali boshqarishingiz mumkin. Agar qo'shimcha savollaringiz bo'lsa, bemalol so'rang! |
| 23. Jirada Scrum board yarating va Sprintni rejalashtiring. Sprintga “issue”larni qo‘shing va Sprint boshlash jarayonini ko‘rsating | Jira'da Scrum board yaratish va Sprint rejalashtirish jarayoni quyidagi bosqichlardan iborat:  ▎1. Scrum Board yaratish  1. Jira'ga kirish: Jira hisobingizga kiring.  2. Loyihangizni tanlang: Yuqori menyudan kerakli loyihani tanlang.  3. Board yaratish:  • Yuqori menyuda "Boards" (Boshqaruvlar) bo'limini tanlang.  • "Create board" (Boshqaruv yaratish) tugmasini bosing.  • "Create a Scrum board" (Scrum boshqaruvini yaratish) variantini tanlang.  • "Create from an existing project" (Mavjud loyihadan yaratish) yoki "Create a new project" (Yangi loyiha yaratish) variantini tanlang, so'ngra kerakli loyihani tanlang.  ▎2. Sprintni rejalashtirish  1. Backlogga o'tish: Yangi yaratilgan Scrum boardda "Backlog" (Orqa reja) bo'limiga o'ting.  2. Muammolarni qo'shish:  • Agar muammolar (issues) hali yaratilmagan bo'lsa, "Create issue" (Muammo yaratish) tugmasini bosing va kerakli muammolarni yarating.  • Yaratilgan muammolarni backlogga qo'shishingiz mumkin.  3. Sprint yaratish:  • Backlog sahifasida yuqori qismda "Create Sprint" (Sprint yaratish) tugmasini bosing.  • Yaratilgan sprint ustiga bosib, kerakli muammolarni (issues) tortib olib qo'shing.  ▎3. Sprintga muammolarni qo'shish  1. Muammolarni tanlash: Backlogda mavjud muammolarni ko'ring va kerakli muammolarni tanlang.  2. Muammolarni qo'shish: Tanlangan muammolarni yangi yaratilgan sprintga torting yoki "Add to Sprint" (Sprintga qo'shish) variantini tanlang.  ▎4. Sprint boshlash  1. Sprintga tayyorgarlik: Sprint uchun barcha muammolar tayyor bo'lganligini tekshiring va ularning holatini "To Do" (Qilishingiz kerak) holatiga o'rnating.  2. Sprintni boshlash:  • Backlog sahifasida yaratilgan sprint ustiga bosing.  • "Start Sprint" (Sprintni boshlash) tugmasini bosing.  • Sprint nomi, davomiyligi (masalan, 2 hafta), va boshqa parametrlarni kiritishingiz mumkin.  • "Start" (Boshlash) tugmasini bosing.  ▎Xulosa  Endi siz Jira'da Scrum board yaratdingiz, sprintni rejalashtirdingiz va kerakli muammolarni sprintga qo'shdingiz. Sprint boshlangandan so'ng, siz jamoangiz bilan birgalikda ishlarni kuzatishingiz va muammolarni hal qilish jarayonini boshqarishingiz mumkin. Agar qo'shimcha savollaringiz bo'lsa, bemalol so'rang! |
| 24. Jirada Kanban board yarating va “issue”larni turli bosqichlarga ko‘chirish orqali ishlov jarayonini tasvirlab bering |  |
| 25. Jirada yangi ish jarayonini yarating yoki mavjud workflow’ni o‘zgartiring. Masalan, “To Do”, “In Progress”, va “Done” holatlarini yaratish va ularni sozlashni ko‘rsating. | Jira'da yangi ish jarayonini yaratganda “To Do”, “In Progress”, va “Done” holatlari har biri o'ziga xos vazifalarga ega. Ushbu holatlar ish jarayonining turli bosqichlarini ifodalaydi va loyiha boshqaruvini samarali amalga oshirishga yordam beradi. Quyida har bir holatning vazifalari keltirilgan:  ▎1. To Do (Bajarilishi kerak)  • Vazifalar:  • Bu holat yangi vazifalar yoki ishlar uchun mo'ljallangan.  • Ish boshlanishidan oldin, barcha vazifalar bu yerda joylashtiriladi.  • Jamoa a'zolari vazifalarni ko'rib chiqishi va ularni bajarishga tayyorlanishi mumkin.  • Vazifalarning ustuvorligini belgilash va kerakli ma'lumotlarni (masalan, tavsif, muddat) qo'shish imkoniyatini beradi.  ▎2. In Progress (Bajaryapti)  • Vazifalar:  • Bu holat ish davom etayotganini ko'rsatadi.  • Jamoa a'zolari ushbu holatda vazifani bajarish jarayonida bo'lishi kerak.  • Ish jarayonida muammolar yoki to'siqlar yuzaga kelganda, jamoa a'zolari bu holatda muhokama olib borishi mumkin.  • Vazifa bajarilayotganligi haqida yangilanishlar kiritish va natijalarni kuzatish imkonini beradi.  ▎3. Done (Bajarildi)  • Vazifalar:  • Bu holat ish tugallanganligini bildiradi.  • Barcha talablar bajarilgandan so'ng, vazifa ushbu holatga o'tkaziladi.  • Tugallangan vazifalar tahlil qilinishi mumkin, bu esa kelgusi ish jarayonlarini yaxshilash uchun foydali bo'ladi.  • Jamoa a'zolari tugallangan ishlar bo'yicha hisobotlar tayyorlash va muvaffaqiyatlarni baholash imkoniyatiga ega bo'ladi.  ▎Xulosa  Ushbu holatlar ish jarayonining samaradorligini oshiradi va jamoaning ishini yaxshiroq boshqarishga yordam beradi. Har bir holat o'z vazifalari bilan birga, jamoa a'zolariga ishni kuzatish va boshqarish imkoniyatini beradi. Agar sizda qo'shimcha savollar bo'lsa yoki boshqa mavzularni muhokama qilishni xohlasangiz, bemalol so'rang! |
| 26. Jirada bitta “issue”ga subtasks yarating. Har bir subtaskni boshqa jamoa a’zolariga taqsimlang va ularning jarayonini kuzating. | Jira'da bitta "issue"ga subtasks yaratish va ularni boshqa jamoa a'zolariga taqsimlash jarayoni quyidagi bosqichlardan iborat:  ▎1. Asosiy "Issue" Yaratish  • Jira'da yangi "issue" (masalan, vazifa yoki loyiha) yarating.  • "Issue"ga kerakli ma'lumotlarni (sarlavha, tavsif, muddat) kiriting.  ▎2. Subtasks Yaratish  • Asosiy "issue" sahifasiga o'ting.  • "Subtasks" bo'limini toping (bu odatda "Issue" sahifasining pastki qismida joylashgan).  • "Create Subtask" tugmasini bosing.  • Har bir subtask uchun kerakli ma'lumotlarni kiriting (sarlavha, tavsif, muddat va boshqalar).  • Har bir subtaskni yaratgandan so'ng, uni saqlang.  ▎3. Subtasks'ni Jamoa A'zolariga Taqsimlash  • Har bir subtaskni oching.  • "Assignee" maydonida subtaskni bajaradigan jamoa a'zosining ismini tanlang.  • O'zgarishlarni saqlang.  ▎4. Jarayonni Kuzatish  • Jira'da "issue" va uning subtasks holatini kuzatib boring.  • Har bir subtask holatini yangilab turish (masalan, "To Do", "In Progress", "Done") orqali jarayonni boshqaring.  • Jamoa a'zolari o'z vazifalarini bajarish jarayonida yangilanishlar kiritishi mumkin.  ▎5. Hisobot va Tahlil  • Tugallangan subtasks'lar bo'yicha hisobot tayyorlang va jamoa a'zolarining ishlarini baholang.  • Jarayon davomida yuzaga kelgan muammolarni muhokama qiling va kelgusi ishlar uchun takliflar tayyorlang.  ▎Xulosa  Bu jarayon yordamida siz Jira'da bitta "issue"ga subtasks yaratishingiz, ularni jamoa a'zolariga taqsimlashingiz va jarayonni samarali ravishda kuzatishingiz mumkin. Agar qo'shimcha savollar yoki yordam kerak bo'lsa, bemalol so'rang! |
| 27. Jirada Sprint tugagach, burndown chartni ko‘rib, Sprint davomida qanday ish bajarilganini tahlil qiling. | Burndown chart — bu Scrum metodologiyasida ishlatiladigan vosita bo'lib, sprint davomida bajarilgan ishlarni vizual ravishda ko'rsatadi. Sprint tugagach burndown chartni tahlil qilish quyidagi bosqichlardan iborat bo'lishi mumkin:  1. Chiziqning ko'rinishi:  • Burndown chartda chiziq qanday ko'rinishda? Agar chiziq to'g'ri pastga tushsa, bu ishlar muvaffaqiyatli bajarilganini ko'rsatadi. Agar chiziq pastga tushish o'rniga tekis yoki ko'tarilsa, bu ishlarning bajarilishida muammolar borligini anglatishi mumkin.  2. Ishning bajarilishi:  • Sprint davomida rejalashtirilgan ishlar (story points yoki tasklar) qanchalik bajarilgan? Agar reja bo'yicha bajarilgan ishlar ko'p bo'lsa, bu jamoaning samaradorligini ko'rsatadi.  3. Muammolar va to'siqlar:  • Sprint davomida qanday muammolar yuzaga keldi? Burndown chartda ishning to'xtab qolgan joylarini aniqlab, jamoa tomonidan qanday to'siqlar mavjudligini tahlil qilish mumkin.  4. O'zgarishlar va kechikishlar:  • Agar burndown chartda ishlar kechikayotgan bo'lsa, bu sabablarini aniqlash zarur. Jamoa a'zolari yoki resurslar bilan bog'liq muammolar bormi?  5. Jamoaning samaradorligi:  • Jamoa o'z maqsadlariga erishdimi? Agar jamoa belgilangan maqsadga erisha olmagan bo'lsa, bu jamoaning samaradorligini oshirish uchun qanday o'zgarishlar kiritilishi kerakligini muhokama qilish zarur.  6. O'rganish va takliflar:  • Tahlil natijalari asosida kelgusi sprintlar uchun qanday o'rganishlar mavjud? Jamoa o'z ish jarayonini yaxshilash uchun qanday takliflar kiritishi mumkin?  Bu tahlil jarayoni jamoaga o'z ish faoliyatini yaxshilash va kelgusi sprintlarda yanada samarali ishlashga yordam beradi. |
| 28. Jirada o‘zingizga xos bo‘lgan moslashtirilgan maydon (custom field) yarating va uni “issue”larga qo‘shing. Masalan, loyiha uchun “Risk darajasi” maydonini qo‘shing. | Jirada moslashtirilgan maydon (custom field) yaratish va uni "issue"larga qo'shish jarayoni quyidagi bosqichlarni o'z ichiga oladi:  ▎1. Moslashtirilgan maydonni yaratish  1. Jira'ga kiring: Jira hisobingizga kirishingiz kerak.  2. Admin paneliga o'ting: Yuqori o'ng burchakda joylashgan "Settings" (Sozlamalar) tugmasini bosing va "Issues" (Masalalar) bo'limiga o'ting.  3. Custom fields bo'limini tanlang: "Fields" bo'limida "Custom fields" (Moslashtirilgan maydonlar) opsiyasini tanlang.  4. Yangi maydon qo'shish: "Add custom field" (Moslashtirilgan maydon qo'shish) tugmasini bosing.  5. Maydon turini tanlang: "Select a field type" (Maydon turini tanlash) oynasida, masalan, "Select List" (Tanlov ro'yxati) yoki "Number Field" (Raqamli maydon) kabi mos turini tanlang.  6. Maydon nomini kiriting: Maydon nomi sifatida "Risk darajasi" ni kiriting va zarur bo'lsa, tavsif qo'shing.  7. Maydonni saqlash: "Create" (Yaratish) tugmasini bosing.  ▎2. Maydonni loyihaga qo'shish  1. Field Configuration (Maydon konfiguratsiyasi) ga o'ting: Yana admin panelida "Field Configurations" (Maydon konfiguratsiyalari) bo'limini tanlang.  2. Loyihani tanlang: O'zingiz yaratgan "Risk darajasi" maydonini qo'shmoqchi bo'lgan loyiha uchun mos keladigan maydon konfiguratsiyasini tanlang.  3. Maydonni qo'shish: "Add field" (Maydon qo'shish) tugmasini bosing va yangi yaratgan "Risk darajasi" maydonini tanlang.  4. Saqlash: O'zgarishlarni saqlang.  ▎3. Moslashtirilgan maydonni “issue”larga qo'shish  1. Loyihangizdagi masalani oching: O'zingiz yaratgan "Risk darajasi" maydonini qo'shmoqchi bo'lgan masalani oching.  2. Masalani tahrirlash: Masalaning yuqori qismidagi "Edit" (Tahrirlash) tugmasini bosing.  3. Risk darajasi maydoni: Yangi qo'shilgan "Risk darajasi" maydonini toping va kerakli qiymatni kiriting (masalan, past, o'rta, yuqori).  4. O'zgarishlarni saqlash: Tahrirlangan masalani saqlang.  ▎4. Yana bir necha maslahat  • Maydonning ko'rinishi: Agar siz ko'p masalalarda "Risk darajasi" maydonini ko'rishni xohlasangiz, uni masala ko'rinishidagi ustunlar ro'yxatiga ham qo'shishingiz mumkin.  • Filtrlar va hisobotlar: Yangi maydondan foydalanib, filtrlar va hisobotlar yaratishingiz mumkin, bu sizga loyiha risklarini yaxshiroq boshqarishga yordam beradi.  Ushbu jarayon orqali siz Jira'da o'z loyihangiz uchun moslashtirilgan "Risk darajasi" maydonini muvaffaqiyatli yaratishingiz va undan foydalanishingiz mumkin. |
| 29. Jirada e-mail yoki Slack kabi vositalar orqali muhim yangiliklar uchun notifikatsiyalarni qanday sozlash mumkinligini ko‘rsating. | Jira'da muhim yangiliklar uchun notifikatsiyalarni sozlash uchun quyidagi bosqichlarni bajaring:  ▎1. Jira'da notifikatsiyalarni sozlash  1. Jira'ga kiring: Jira hisobingizga kirishingiz kerak.  2. Admin paneliga o'ting: Yuqori o'ng burchakda joylashgan "Settings" (Sozlamalar) tugmasini bosing.  3. System bo'limiga o'ting: "Issues" bo'limidan "Notification schemes" (Notifikatsiya sxemalari) opsiyasini tanlang.  4. Notifikatsiya sxemasini tanlang: Siz ishlatmoqchi bo'lgan notifikatsiya sxemasini tanlang yoki yangi sxema yarating.  5. Notifikatsiyalarni qo'shish: "Add" (Qo'shish) tugmasini bosib, muhim voqealar (masalan, masala yaratish, yangilash, yopish va h.k.) uchun kimlarga notifikatsiya yuborilishi kerakligini tanlang. Bu yerda foydalanuvchilar, guruhlar yoki boshqa ro'yxatlar tanlanishi mumkin.  6. O'zgarishlarni saqlash: O'zgartirishlarni saqlang.  ▎2. E-mail orqali notifikatsiyalarni sozlash  1. E-mail konfiguratsiyasi: Admin panelida "System" bo'limida "Email notifications" (E-mail notifikatsiyalari) opsiyasini tanlang va kerakli parametrlarni sozlang.  2. Foydalanuvchilarni qo'shish: Notifikatsiya sxemasida e-mail orqali xabar olishni xohlagan foydalanuvchilarni qo'shing.  3. Sinov: E-mail notifikatsiyalarini sinab ko'ring, shunda sizning o'zgarishlaringiz to'g'ri ishlayotganini tasdiqlashingiz mumkin.  ▎3. Slack orqali notifikatsiyalarni sozlash  1. Jira va Slack integratsiyasini o'rnatish:  • Jira'da "Apps" (Dasturlar) bo'limiga o'ting va "Find new apps" (Yangi dasturlarni topish) opsiyasini tanlang.  • "Slack" dasturini qidiring va uni o'rnating.    2. Slack kanali bilan bog'lanish:  • O'rnatilgan Slack integratsiyasidan foydalanib, Jira'dagi yangiliklar uchun ma'lum bir Slack kanalini tanlang.  • Integratsiya jarayonida sizdan Slack hisobingizga kirishingiz va kerakli kanalni tanlashingiz so'raladi.  3. Notifikatsiya sozlamalarini tanlash:  • Jira'da Slack integratsiyasi orqali qanday voqealar uchun notifikatsiya yuborilishini tanlang (masalan, masala yaratish, yangilash, yopish va h.k.).  • O'zgarishlarni saqlang.  ▎4. Notifikatsiyalarni boshqarish  • Personalizatsiya: Har bir foydalanuvchi o'z profilida notifikatsiya sozlamalarini shaxsiylashtirishi mumkin. Buning uchun "Profile" (Profil) bo'limiga o'ting va "Notifications" (Notifikatsiyalar) parametrlarini tahrirlang.  • Qidiruv va filtrlar: Jira'da qidiruv va filtrlar yordamida muhim masalalarni kuzatishingiz mumkin.  Ushbu bosqichlar orqali siz Jira'da muhim yangiliklar uchun e-mail yoki Slack orqali notifikatsiyalarni muvaffaqiyatli sozlashingiz mumkin. |
| 30. Jirada ikkita “issue” o‘rtasida bog‘lanishni yarating (masalan, “block”, “relates to”). Bu bog‘lanish qanday ta’sir ko‘rsatishini ko‘rsating. | Jira'da ikkita "issue" o'rtasida bog'lanishni yaratish, masalalar o'rtasidagi munosabatlarni aniq belgilashga yordam beradi. Bu bog'lanishlar loyihani boshqarish jarayonida muhim ahamiyatga ega, chunki ular masalalarni bir-biri bilan bog'lash va ularning o'zaro ta'sirini ko'rsatishga imkon beradi. Keling, bog'lanishlarni qanday yaratish va ularning ta'sirini ko'rib chiqamiz.  ▎Bog'lanishni yaratish  1. Masalalarni tanlash: Bog'lanishni yaratmoqchi bo'lgan ikkita masalani tanlang. Masalan, "Issue A" va "Issue B".  2. Masala sahifasiga o'ting: "Issue A" masalasini oching.  3. Linked Issues (Bog'langan masalalar) bo'limini toping: Masala sahifasining pastki qismida "Linked Issues" bo'limini toping.  4. Bog'lanish qo'shish: "Link" (Bog'lanish) tugmasini bosing.  5. Bog'lanish turini tanlang: Ochiq oynada bog'lanish turini tanlang (masalan, "blocks", "relates to", "is blocked by" va h.k.).  6. Ikkinchi masalani kiriting: "Issue B" ning identifikatorini yoki nomini kiriting va uni tanlang.  7. Saqlash: O'zgarishlarni saqlang.  ▎Bog'lanishning ta'siri  1. O'zaro bog'liqlik: Masalalar o'rtasidagi bog'lanishlar loyihani boshqarishda muhim rol o'ynaydi. Masalan, agar "Issue A" "Issue B" ni bloklayotgan bo'lsa, bu "Issue B" ni yechmasdan "Issue A" ni hal qilish mumkin emasligini anglatadi.  2. Kuzatuvchanlik: Bog'langan masalalar o'rtasida kuzatuvchanlikni oshiradi. Agar bir masala yangilansa yoki o'zgarishlar kiritilsa, boshqa bog'langan masalalar ham yangilanadi va bu orqali jamoa a'zolari bir-birining ishlarini yaxshiroq tushunishi mumkin.  3. Hisobotlar: Jira'da masalalar o'rtasidagi bog'lanishlar hisobotlarda ko'rinishi mumkin. Bu loyiha boshqaruvchilarga masalalarni tahlil qilishda va resurslarni rejalashtirishda yordam beradi.  4. Ish jarayonini optimallashtirish: Bog'lanishlar ish jarayonini optimallashtirishga yordam beradi, chunki jamoa a'zolari bir-birining ishlariga mos ravishda harakat qilishlari mumkin.  5. Prioritizatsiya: Masalalar o'rtasidagi bog'lanishlar yordamida jamoa a'zolari qaysi masalalar ustuvor ekanligini aniqlashlari mumkin. Masalan, agar bir masala boshqa bir masala tomonidan bloklanayotgan bo'lsa, u holda birinchi masalani hal qilish zaruriyati paydo bo'ladi.  ▎Xulosa  Jira'da masalalar o'rtasida bog'lanishni yaratish orqali siz loyiha boshqaruvini yanada samarali amalga oshirishingiz mumkin. Bu bog'lanishlar masalalar o'rtasidagi o'zaro ta'sirni ko'rsatadi va jamoaga ish jarayonini yaxshiroq boshqarishga yordam beradi. |
| 31. Jirani Confluence bilan integratsiya qilib, Jira “issue”larini Confluence hujjatlarida qanday ko‘rsatishni ko‘rsating. | Jira va Confluence integratsiyasi orqali Jira masalalarini Confluence hujjatlarida ko'rsatish juda qulay va samarali usuldir. Bu jarayon sizga loyihalar va ish jarayonlarini yanada samarali boshqarishga yordam beradi. Quyida Jira “issue”larini Confluence hujjatlarida qanday ko'rsatishni ko'rsatib beraman:  ▎1. Integratsiyani o'rnatish  Avvaliga, Jira va Confluence o'rtasidagi integratsiyani o'rnatishingiz kerak. Agar sizda bu integratsiya allaqachon o'rnatilgan bo'lsa, ushbu qadamni o'tkazishingiz mumkin.  1. Confluence va Jira administratorlari bilan bog'laning: Ikkala platformada ham administratorlar bo'lishi kerak.  2. Integratsiya sozlamalarini tekshiring: Confluence'da "Settings" (Sozlamalar) > "Application links" (Ilova havolalari) bo'limiga o'ting va Jira ilovasi mavjudligini tekshiring. Agar yo'q bo'lsa, yangi havola qo'shing.  ▎2. Jira masalalarini Confluence hujjatiga qo'shish  1. Confluence hujjatini oching: Jira masalalarini qo'shmoqchi bo'lgan Confluence sahifasini oching yoki yangi sahifa yarating.    2. Makro qo'shish:  • Sahifaning tahrir rejimida bo'lganingizda, "Insert more content" (Boshqa mazmun qo'shish) tugmasini bosing (yoki "+" belgisini bosing).  • "Other macros" (Boshqa makrolar) bo'limini tanlang.  3. Jira makrosini tanlang:  • Ochiq oynada "Jira" deb qidiring va "Jira Issue/Filter" makrosini tanlang.  4. Masalani tanlash:  • Oynada sizga bir nechta variantlar taqdim etiladi:  • Issue key: Muayyan masalaning kalitini (masalan, PROJ-123) kiriting.  • JQL query: Agar bir nechta masalalarni ko'rsatmoqchi bo'lsangiz, JQL (Jira Query Language) so'rovini yozing.  • Filter: Agar sizda oldindan belgilangan filtr bo'lsa, uni tanlashingiz mumkin.    5. Ko'rsatish parametrlarini sozlash:  • Masalalarni qanday ko'rsatishni tanlang (masalan, ustunlar, holat va boshqalar).  • O'zgarishlarni tasdiqlash uchun "Insert" (Qo'shish) tugmasini bosing.  6. Sahifani saqlang:  • Hujjatni saqlang va natijani ko'ring. Jira masalalari Confluence sahifasida ko'rsatiladi.  ▎3. Masalalarni yangilash  Agar siz Jira'da masalalarni yangilasangiz, Confluence sahifasidagi ma'lumotlar avtomatik ravishda yangilanadi. Bu sizga har doim eng so'nggi ma'lumotlarga ega bo'lish imkonini beradi.  ▎4. Qo'shimcha imkoniyatlar  • Dashbordlar yaratish: Confluence sahifalarida Jira ma'lumotlarini ko'rsatadigan maxsus dashbordlar yaratishingiz mumkin.  • Grafiklar va diagrammalar: Jira ma'lumotlarini tahlil qilish uchun grafiklar va diagrammalar qo'shishingiz mumkin.  ▎Xulosa  Jira masalalarini Confluence hujjatlarida ko'rsatish orqali siz jamoangizning ish jarayonlarini yanada samarali boshqarishingiz mumkin. Bu integratsiya loyihalarni kuzatishda, hisobotlar tayyorlashda va jamoa a'zolari o'rtasida aloqa o'rnatishda katta yordam beradi. |
| 32. Jirada loyiha a’zolariga turli ruxsatlar bering (masalan, “view”, “edit”, “delete” huquqlarini sozlang) va bu qanday ishlashini ko‘rsating. | Jira'da loyiha a'zolariga turli ruxsatlar berish, loyiha boshqaruvi va jamoa ishini samarali amalga oshirish uchun muhim ahamiyatga ega. Ruxsatlar orqali siz har bir a'zoning loyihadagi roliga va vazifasiga mos ravishda huquqlarini belgilashingiz mumkin. Quyida Jira'da loyiha a'zolariga ruxsatlarni qanday berish va sozlash jarayonini ko'rsataman.  ▎1. Loyiha Ruxsatlarini Boshqarish  ▎a. Ruxsatlar to'g'risida tushuncha  Jira'da ruxsatlar asosan uchta asosiy kategoriya bo'yicha tasniflanadi:  • View: Masalalarni ko'rish huquqi.  • Edit: Masalalarni tahrirlash huquqi.  • Delete: Masalalarni o'chirish huquqi.  ▎b. Ruxsatlar to'plamini yaratish  1. Jira’ni oching: Jira hisobingizga kiring.  2. Admin paneliga o'ting: Yuqori o'ng burchakdagi "Settings" (Sozlamalar) tugmasini bosing va "Projects" (Loyihalar) bo'limiga o'ting.  3. Ruxsatlar to'plamini tanlang: O'ng tomondagi menyuda "Permission schemes" (Ruxsatlar sxemalari) ni tanlang.  4. Yangi ruxsatlar to'plamini yarating: "Add Permission Scheme" (Ruxsatlar sxemasini qo'shish) tugmasini bosing va yangi sxemani nomlang  ▎2. Ruxsatlarni belgilash  1. Yangi sxemani tahrirlash: Yaratilgan ruxsatlar sxemasini tanlang va "Permissions" (Ruxsatlar) bo'limiga o'ting.  2. Ruxsatlarni qo'shish:  • "Add Permission" (Ruxsat qo'shish) tugmasini bosing.  • Ruxsatlar ro'yxatidan kerakli ruxsatni tanlang (masalan, "Browse Projects", "Edit Issues", "Delete Issues").  • Ruxsatni kimga berishni tanlang:  • Groups: Guruhga ruxsat berish.  • Project Roles: Loyihadagi rollarga ruxsat berish (masalan, Developer, Tester)  • Single User: Faqat bitta foydalanuvchiga ruxsat berish.  3. Ruxsatlarni saqlash: Har bir o'zgarishni saqlang.  ▎3. Loyiha a'zolariga ruxsatlarni qo'llash  1. Loyihani tanlang: Admin panelida "Projects" (Loyihalar) bo'limiga o'ting va kerakli loyihani tanlang.  2. Ruxsatlar sxemasini qo'llang:  • Loyihani tahrir qilish uchun "Project settings" (Loyiha sozlamalari) ga o'ting.  • "Permissions" (Ruxsatlar) bo'limini tanlang.  • Yangi yaratgan ruxsatlar sxemasini tanlang va saqlang.  ▎4. Ruxsatlarni tekshirish  1. Loyihaga kirish: Loyihaga a'zolar sifatida kirib, ularning ruxsatlarini tekshiring.  2. Masalalarni ko'rish, tahrirlash va o'chirish: Har bir a'zoning masalalarni ko'rishi, tahrirlashi yoki o'chirishi mumkinligini tekshiring.  ▎5. Ruxsatlar haqida eslatmalar  • Ruxsatlar rollari: Har bir loyiha uchun alohida rollar yaratish va ularga mos ravishda ruxsatlarni belgilash yaxshi amaliyotdir.  • Ish jarayonlari: Ruxsatlarni belgilashda loyiha ish jarayonlarini hisobga olish muhimdir, chunki ba'zi a'zolar faqat ma'lum vazifalarni bajarishi kerak.  ▎Xulosa  Jira'da loyiha a'zolariga turli ruxsatlarni berish orqali siz jamoangizning ish jarayonlarini samarali boshqarishingiz mumkin. Bu nafaqat xavfsizlikni ta'minlaydi, balki har bir a'zoning vazifalarini aniq belgilashga yordam beradi. Ruxsatlarni to'g'ri belgilash orqali siz loyihaning muvaffaqiyatini oshirishingiz mumkin. |
| 33. Jirada loyiha yoki Sprint uchun hisobot (report) yarating. Hisobotni tahlil qilish uchun foydalaning va natijalarni muhokama qiling. | Jira'da loyiha yoki sprint uchun hisobot yaratish va uni tahlil qilish jarayoni bir nechta bosqichlardan iborat. Hisobotlar jamoaning ish faoliyatini kuzatish, muammolarni aniqlash va kelajakdagi rejalashtirish uchun juda muhimdir. Quyida loyiha yoki sprint uchun hisobot yaratish va uni tahlil qilish jarayonini ko'rib chiqamiz.  ▎1. Hisobotni yaratish  ▎a. Sprint hisobotini yaratish  1. Jira'ga kirish: Jira hisobingizga kiring.  2. Loyihani tanlash: Yuqori menyudan kerakli loyihani tanlang.  3. Sprintlar bo'limiga o'ting: "Active sprints" (Faol sprintlar) bo'limiga o'ting.  4. Sprint hisobotini oching: O'ng tomondagi "Reports" (Hisobotlar) bo'limiga o'ting va "Sprint Report" (Sprint hisobot) ni tanlang.  ▎b. Hisobot parametrlarini belgilash  1. Sprintni tanlash: O'tgan yoki faol sprintni tanlang.  2. Hisobotni sozlash: Hisobotda ko'rinadigan masalalar, ularning holati (yangi, bajarilgan, bajarilmagan) va vaqt sarfini ko'rsatish uchun kerakli parametrlarni tanlang.  ▎2. Hisobotni tahlil qilish  ▎a. Hisobotdan olingan ma'lumotlar  • Bajarilgan masalalar: Sprint davomida bajarilgan masalalar soni.  • Bajarilmagan masalalar: Sprint tugagach bajarilmagan masalalar soni.  • Yangi masalalar: Sprint davomida qo'shilgan yangi masalalar soni.  • O'tkazilgan ishlar: O'tgan sprintda bajarilishi rejalashtirilgan, lekin bajarilmagan ishlar.  ▎b. Tahlil qilish  1. Bajarilgan va bajarilmagan ishlar:  • Agar bajarilgan ishlar ko'p bo'lsa, bu jamoaning samaradorligini ko'rsatadi.  • Bajarilmagan ishlar soni yuqori bo'lsa, sabablarini aniqlash zarur (masalan, vaqt yetishmasligi, resurslar yetishmasligi yoki noaniq talablar).  2. Yangi masalalar:  • Agar sprint davomida yangi masalalar ko'p qo'shilsa, bu jamoaning ish yukini oshirishi mumkin. Bunday holatlarda loyiha boshqaruvchilari jamoaning yukini baholashi va kelgusi sprintlarga rejalashtirishni yaxshilashi kerak.  3. Vaqt sarfi:  • Har bir masala uchun sarflangan vaqtni tahlil qilish orqali qaysi vazifalar ko'p vaqt talab qilayotganini aniqlash mumkin. Bu muammolarni aniqlash va ularni hal qilish uchun yordam beradi.  ▎3. Natijalarni muokama qilish  ▎a. Jamoa yig'ilishi  Hisobot natijalarini muhokama qilish uchun jamoa yig'ilishini o'tkazish tavsiya etiladi:  1. Yig'ilishga tayyorgarlik:  • Hisobotdagi asosiy ma'lumotlarni tayyorlang.  • Qanday masalalar yaxshi bajarilganligini va qaysilari muammoga olib kelganligini aniqlang.  2. Muhokama qilish:  • Jamoa a'zolari bilan bajarilgan ishlar va muammolar haqida fikr almashing.  • O'tgan sprintda qanday o'zgarishlar kiritilishi mumkinligini muhokama qiling.  3. Kelgusi reja:  • Kelgusi sprintlar uchun qanday maqsadlar qo'yilishi kerakligini belgilash.  • Muammolarni hal qilish va samaradorlikni oshirish uchun strategiyalar ishlab chiqish.  ▎Xulosa  Jira'da loyiha yoki sprint hisobotlarini yaratish va tahlil qilish jarayoni jamoaning ish faoliyatini yaxshilashga yordam beradi. Hisobotlarni muntazam ravishda tayyorlash va tahlil qilish orqali jamoa o'z kuchli va zaif tomonlarini aniqlashi, shuningdek, kelgusi ishlarni rejalashtirishi mumkin. Ushbu jarayon jamoaning samaradorligini oshirish va loyiha muvaffaqiyatini ta'minlash uchun muhimdir. |
| 34. Jirada avtomatlashtirishni sozlang, masalan, muammo holati “done” bo‘lganda avtomatik ravishda subtasks ham yopilsin. | Jira'da avtomatlashtirishni sozlash orqali muammo holati "Done" ga o'zgarganda, uning subtasks (kichik vazifalar) ham avtomatik ravishda yopilishini ta'minlash mumkin. Buning uchun siz quyidagi qadamlarni bajarishingiz kerak:  ▎1. Jira'da Avtomatlashtirish Qoidasi Yaratish  1. Jira'ga kirish: Jira hisobingizga kiring.  2. Loyihani tanlang: O'zgartirmoqchi bo'lgan loyihani tanlang.  3. Avtomatlashtirish bo'limiga o'ting:  • Loyihangizning menyusidan "Project settings" (Loyiha sozlamalari) ga o'ting.  • "Automation" (Avtomatlashtirish) bo'limini tanlang.  4. Yangi qoidani yaratish:  • "Create rule" (Qoidani yaratish) tugmasini bosing.  ▎2. Qoidani Sozlash  ▎a. Trigger (Tugma)  1. Triggerni tanlang:  • "Issue transitioned" (Muammo o'tkazilganda) triggerini tanlang.  • Bu trigger muammo holati o'zgarganda ishga tushadi.  2. Holatni tanlang:  • "From status" (Holatdan) qismida barcha holatlarni tanlang va "To status" (Holatga) qismida "Done" ni tanlang.  ▎b. Action (Harakat)  1. Action qo'shish  • "New action" (Yangi harakat) tugmasini bosing.  • "Edit issue" (Muammo tahrirlash) harakatini tanlang.  2. Subtasks'larni yopish:  • "Choose fields to set..." (O'rnatilishi kerak bo'lgan maydonlarni tanlang...) qismida "Sub-tasks" maydonini tanlang.  • "Status" maydonini tanlang va "Done" ni belgilang.  ▎3. Qoidani Saqlash  1. Qoidani saqlash:  • Qoidaga nom bering, masalan, "Close Subtasks When Parent Done".  • "Save" (Saqlash) tugmasini bosing.  2. Qoidani faollashtirish:  • Qoidani faollashtirish uchun "Turn on" (Yoqing) tugmasini bosing.  ▎4. Qoidani Sinovdan O'tkazish  1. Sinov muammosini yarating: Biror muammo yarating va unga subtasks qo'shing.  2. Muammo holatini o'zgartiring: Muammo holatini "Done" ga o'zgartiring.  3. Natijalarni tekshiring: Avtomatlashtirish qoidasi to'g'ri ishlaganligini tekshirib ko'ring — subtasks holati ham avtomatik ravishda "Done" ga o'zgarishi kerak.  ▎Xulosa  Ushbu qadamlar yordamida siz Jira'da muammo holati "Done" ga o'zgarganda avtomatik ravishda uning subtasks'larini ham yopadigan avtomatlashtirish qoidasini yaratishingiz mumkin. Bu jarayon ish jarayonini soddalashtirish va samaradorlikni oshirishga yordam beradi. |
| 35. GitHub repositorysidan foydalanib, GitHub Pagesda oddiy statik veb-sayt yarating va uni GitHubdagi repository ga ulab qo‘ying. | GitHub Pages yordamida oddiy statik veb-sayt yaratish va uni GitHub repository'siga ulash uchun quyidagi qadamlarni bajaring:  ▎1. GitHub'da Repository Yaratish  1. GitHub hisobingizga kiring.  2. Yangi repository yarating:  • O'ng yuqoridagi "+" tugmasini bosing va "New repository" ni tanlang.  • Repository nomini kiriting (masalan, my-website).  • "Public" yoki "Private" variantini tanlang.  • "Initialize this repository with a README" variantini tanlang (agar xohlasangiz).  • "Create repository" tugmasini bosing.  ▎2. Veb-sayt Fayllarini Yaratish  1. Lokal kompyuteringizda yangi papka yarating:    mkdir my-website  cd my-website    2. HTML faylini yarating:  • index.html nomli fayl yarating va unda quyidagi kodni joylashtiring:    <!DOCTYPE html>  <html lang="uz">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>My GitHub Pages Site</title>  </head>  <body>  <h1>Salom, dunyo!</h1>  <p>Bu mening birinchi GitHub Pages saytimm.</p>  </body>  </html>    ▎3. Fayllarni GitHub Repository'siga Qo'shish  1. Gitni boshlang:    git init    2. Repository'ni qo'shing:    git remote add origin https://github.com/username/my-website.git    username o'rniga GitHub foydalanuvchi nomingizni yozing.  3. Fayllarni qo'shing va commit qiling:    git add .  git commit -m "First commit"  4. O'zgarishlarni GitHub'ga yuklang:  git push -u origin main  ▎4. GitHub Pages'ni Yoqing  1. Repository sahifasiga o'ting: GitHub'dagi yangi yaratgan repository sahifasiga o'ting.  2. Settings (Sozlamalar) bo'limiga o'ting: Yuqori o'ng burchakdagi "Settings" tugmasini bosing.  3. Pages bo'limiga o'ting: Chap tomonda "Pages" bo'limini tanlang.  4. Branch tanlash: "Source" qismida "main" branchni tanlang va "/ (root)" variantini belgilang.  5. Saqlash: O'zgarishlarni saqlang.  ▎5. Saytingizga Kirish  1. Sayt manzilingizni toping: GitHub Pages sayt manzilingiz quyidagi formatda bo'ladi:  https://username.github.io/my-website/  username o'rniga GitHub foydalanuvchi nomingizni yozing.  ▎Xulosa  Endi siz GitHub Pages yordamida oddiy statik veb-sayt yaratdingiz va uni GitHub repository'siga uladingiz! Saytingizga kirib ko'rishingiz mumkin va kelajakda uni yanada rivojlantirish uchun yangi sahifalar va kontent qo'shishingiz mumkin. |
| 36. GitHub repository’sida yangi “issue” yarating va uni loyihaga tayinlang. Muammoni hal qilib, uni yopib qo‘ying. | GitHub repository'sida yangi "issue" yaratish, uni loyihaga tayinlash, muammoni hal qilish va uni yopish jarayoni quyidagi qadamlar orqali amalga oshiriladi:  ▎1. Yangi "Issue" Yaratish  1. GitHub hisobingizga kiring va o'z repository'ingizga o'ting.  2. "Issues" bo'limiga o'ting: Yuqori menyuda "Issues" tabini tanlang.  3. Yangi issue yarating:  • O'ng yuqoridagi "New issue" tugmasini bosing.  • Issue sarlavhasini (masalan, "Sayt dizaynini yangilash") va batafsil tavsifini kiriting.  • Agar kerak bo'lsa, teglar (labels) va boshqa parametrlarni qo'shing.  4. "Submit new issue" tugmasini bosing.  ▎2. "Issue"ni Loyihaga Tayinlash  1. Yaratilgan issue sahifasida, o'ng tomonda "Projects" bo'limini toping.  2. "Projects" bo'limida, kerakli loyihani tanlang (agar loyihalar oldin yaratilgan bo'lsa).  3. Tayinlash uchun "Add to project" yoki shunga o'xshash tugmani bosing va loyihani tanlang.  ▎3. Muammoni Hal Qilish  1. Muammoni hal qilish: Muammoni hal qilish uchun kodni tahrir qiling va yangi commit qiling.  • O'zgartirishlarni amalga oshiring va commit qiling:    git add .  git commit -m "Muammoni hal qildim"  git push  ▎4. "Issue"ni Yopish  1. Yana issue sahifasiga qayting.  2. Hal qilingan issue ni toping va oching.  3. O'ng tomonda "Close issue" tugmasini bosing.  ▎Xulosa  Endi siz GitHub repository'sida yangi issue yaratdingiz, uni loyihaga tayinladingiz, muammoni hal qildingiz va uni yopdingiz! Bu jarayon sizga muammolarni boshqarishda va jamoa bilan ishlashda yordam beradi. |
| 37. GitHubda yangi pull request yarating va boshqa jamoa a’zosiga ko‘rib chiqish uchun yuboring. Pull requestning tahlil jarayonini tushuntirib bering. | GitHub'da yangi pull request (PR) yaratish va uni boshqa jamoa a'zosiga ko'rib chiqish uchun yuborish jarayoni quyidagi qadamlar orqali amalga oshiriladi:  ▎1. Pull Request Yaratish  1. O'z repository'ingizga kiring: GitHub'dagi o'z repository'ingizga o'ting.  2. Branch yaratish: O'zgarishlarni amalga oshirishdan oldin yangi branch yarating.  git checkout -b yangi-branch-nomi    3. O'zgarishlarni amalga oshiring: Kodda kerakli o'zgarishlarni bajaring va commit qiling.    git add .  git commit -m "O'zgarishlar qo'shildi"  4. Branchni push qiling: O'zgarishlarni GitHub'ga yuklang.  git push origin yangi-branch-nomi  5. Pull request yarating:  • GitHub'da repository sahifasiga qaytib, "Pull requests" tabini tanlang.  • O'ng yuqoridagi "New pull request" tugmasini bosing.  • Yangi branchni asosiy branch (masalan, main yoki master) bilan solishtiring.  • O'zgarishlaringizni ko'ring va "Create pull request" tugmasini bosing.  • Pull request sarlavhasini va batafsil tavsifini kiriting. Agar kerak bo'lsa, teglar va tayinlangan shaxslarni qo'shing.  • "Create pull request" tugmasini bosing.  ▎2. Pull Requestni Ko'rib Chiqish Jarayoni  1. Ko'rib chiqish jarayoni:  • Pull request yaratgandan so'ng, tayinlangan jamoa a'zosi (reviewer) sizning o'zgarishlaringizni ko'rib chiqadi.  • Reviewer, kodni o'qiydi, o'zgarishlarni sinab ko'radi va kerak bo'lsa, izohlar qoldiradi.  2. Izohlar va muhokama:  • Reviewer izohlar qoldirishi mumkin, masalan, kodda yaxshilanishlarni taklif qilish yoki muammolarni ko'rsatishi mumkin.  • Siz ushbu izohlar asosida kodni yangilashingiz mumkin. Buning uchun o'z branchingizda o'zgarishlar qiling va yana commit qiling:  git add .  git commit -m "Izohlar asosida o'zgartirishlar kiritildi"  git push origin yangi-branch-nomi  3. Pull requestni tasdiqlash:  • Reviewer sizning o'zgarishlaringizni ko'rib chiqib, agar hamma narsa yaxshi bo'lsa, "Approve" tugmasini bosishi mumkin.  • Tasdiqlashdan so'ng, siz yoki boshqa jamoa a'zosi pull requestni birlashtirish (merge) jarayonini amalga oshirishi mumkin.  4. Pull requestni birlashtirish:  • Pull request sahifasida "Merge pull request" tugmasini bosib, o'zgarishlarni asosiy branchga birlashtiring.  • Yana bir bor tasdiqlash uchun "Confirm merge" tugmasini bosing.  5. Pull requestni yopish:  • O'zgarishlar birlashtirilgandan so'ng, pull request avtomatik ravishda yopiladi yoki siz uni qo'lda yopishingiz mumkin.  ▎Xulosa  Endi siz GitHub'da yangi pull request yaratdingiz va uni boshqa jamoa a'zosiga ko'rib chiqish uchun yubordingiz. Pull request jarayoni jamoaviy ishlashda muhim ahamiyatga ega bo'lib, kod sifatini ta'minlash va muammolarni hal qilishda yordam beradi. |
| 38. GitHub’da yangi branch yarating, ushbu branchda o‘zgartirishlar kiriting va keyin asosiy branch bilan birlashtiring (merge). Merge konfliktlari bo‘lsa, ularni qanday hal qilasiz? | GitHub’da yangi branch yaratish, o‘zgarishlar kiritish va keyin asosiy branch bilan birlashtirish jarayonini quyidagi qadamlar orqali amalga oshirishingiz mumkin. Agar merge konfliktlari yuzaga kelsa, ularni qanday hal qilishni ham ko‘rsataman.  ▎1. Yangi Branch Yaratish  1. Terminalni oching va o‘z repository’ingizga o‘ting:    cd /path/to/your/repo    2. Asosiy branchga o‘ting (odatan main yoki master):    git checkout main    3. Yangi branch yarating:    git checkout -b yangi-branch-nomi    ▎2. O‘zgarishlarni Kiritish  1. Kodda o‘zgarishlar qiling: O‘zingiz xohlagan fayllarni tahrir qiling.  2. O‘zgarishlarni qo‘shing va commit qiling:    git add .  git commit -m "O'zgarishlar kiritildi"    3. Branchni GitHub’ga push qiling:    git push origin yangi-branch-nomi    ▎3. Asosiy Branch Bilan Birlashtirish (Merge)  1. Asosiy branchga qayting:    git checkout main    2. Asosiy branchni yangilang (agar boshqa jamoa a'zolari o‘zgarishlar kiritgan bo‘lsa):    git pull origin main    3. Yangi branchni asosiy branch bilan birlashtiring:    git merge yangi-branch-nomi    ▎4. Merge Konfliktlarini Hal Qilish  Agar merge jarayonida konfliktlar yuzaga kelsa, quyidagi qadamlarni bajaring:  1. Konfliktlarni aniqlang: Git sizga konflikt bo‘lgan fayllarni ko‘rsatadi. Ular odatda CONFLICT deb belgilanadi.  2. Konfliktlarni hal qiling:  • Konflikt bo‘lgan faylni oching va <<<<<<<, =======, va >>>>>>> belgilari yordamida qaysi qismni saqlashni tanlang.  • O‘zgarishlarni qo‘shing yoki o‘zgartiring va faylni saqlang.  3. Faylni qo‘shing va commit qiling:    git add fayl-nomi  git commit -m "Konfliktlar hal qilindi"    4. Birlashtirish jarayonini tugatish: Agar siz boshqa branchlarga ham o‘zgartirishlar kiritgan bo‘lsangiz, ularni birlashtirish jarayonini yakunlash uchun quyidagi buyruqni bajaring:    git merge --continue    ▎5. O‘zgarishlarni GitHub’ga Push Qilish  1. O‘zgarishlarni GitHub’ga yuklang:    git push origin main  ▎Xulosa  Endi siz GitHub’da yangi branch yaratdingiz, o‘zgarishlar kiritdingiz va asosiy branch bilan birlashtirdingiz. Agar merge konfliktlari yuzaga kelsa, ularni hal qilish jarayonini ham o‘rganib oldingiz. Bu jarayon jamoaviy ishda muhim ahamiyatga ega va kod sifatini ta'minlashda yordam beradi. |
| 39. GitHubda yangi repository yarating. Mahalliy kompyuterdagi loyihangizni ushbu repository ga yuklang (git push) va uni ochiq (public) yoki yopiq (private) qilib sozlang. | GitHub’da yangi repository yaratish va mahalliy kompyuterdagi loyihangizni ushbu repository ga yuklash jarayoni quyidagi qadamlar orqali amalga oshiriladi. Ushbu jarayonni bajarish uchun sizda Git o‘rnatilgan bo‘lishi va GitHub hisobingiz bo‘lishi kerak.  ▎1. GitHub’da Yangi Repository Yaratish  1. GitHub hisobingizga kiring.  2. "Repositories" bo‘limiga o‘ting va "New" tugmasini bosing.  3. Repository nomini kiriting: Bu sizning loyihangizning nomi bo‘ladi.  4. Description (ixtiyoriy): Loyihangiz haqida qisqacha ma'lumot kiriting.  5. Visibility:  • Public: Har kim ko‘ra oladi.  • Private: Faqat siz va siz ruxsat bergan foydalanuvchilar ko‘ra oladi.  6. "Initialize this repository with a README" opsiyasini tanlamang, agar siz allaqachon mahalliy repositoryga ega bo‘lsangiz.  7. "Create repository" tugmasini bosing.  ▎2. Mahalliy Loyihangizni GitHub Repositorysiga Yuklash  ▎2.1. Mahalliy Repositoryni Tayyorlash  Agar sizda hali mahalliy repository bo‘lmasa, quyidagi qadamlarni bajaring:  1. Terminalni oching va loyiha papkangizga o‘ting:    cd /path/to/your/project    2. Git repository yarating:    git init    3. Barcha fayllarni qo‘shing:  git add .  4. Birinchi commitni amalga oshiring:  git commit -m "Birinchi commit"  ▎2.2. Yangi Repositoryga Push Qilish  1. Yangi repository manzilini qo‘shing:  git remote add origin https://github.com/username/repository-name.git  username o‘rniga GitHub username’ingizni va repository-name o‘rniga yangi repository nomini yozing.  2. Mahalliy o‘zgarishlarni GitHub’ga yuklang:  git push -u origin main  Agar sizning asosiy branchingiz master bo‘lsa, main o‘rniga master yozing.  ▎3. Repository Sozlamalarini Tekshirish  1. GitHub sahifasiga qayting va yangi yaratgan repository’ingizga o‘ting.  2. Settings bo‘limiga o‘ting (repository sahifasining yuqori qismidagi tab).  3. Danger Zone bo‘limida repository’ning xususiyatlarini o‘zgartirishingiz mumkin, masalan, uni public yoki private qilib sozlashingiz mumkin.  ▎Xulosa  Endi siz GitHub’da yangi repository yaratdingiz va mahalliy loyihangizni ushbu repository ga muvaffaqiyatli yukladingiz. Agar kerak bo‘lsa, repository’ning xususiyatlarini (public yoki private) o‘zgartirishingiz mumkin. Agar qo‘shimcha savollaringiz bo‘lsa, bemalol so‘rashingiz mumkin! |
| 40. Reinforcement learning algoritmidan foydalangan holda, agentga ma'lum bir vazifani bajarishni o‘rgating (masalan, ma'lum bir harakatni mukofotlash orqali). Agent qanday qilib o‘z strategiyasini rivojlantiradi? | Reinforcement Learning (RL) — bu agentning muhit bilan o'zaro aloqasini o'rganish va mukofotlar orqali strategiyalarini rivojlantirish jarayoni. Agent ma'lum bir vazifani bajarishni o'rganish uchun quyidagi asosiy tushunchalardan foydalanadi:  ▎1. Asosiy Tushunchalar  • Agent: O‘rganayotgan subyekt (masalan, robot, dastur).  • Muhit (Environment): Agent o‘zaro aloqada bo‘lgan tashqi dunyo.  • Holat (State): Agentning muhitdagi joriy holati.  • Harakat (Action): Agentning muhitda bajarishi mumkin bo‘lgan harakatlari.  • Mukofot (Reward): Agentning harakatidan so‘ng oladigan natija; agentni ma'lum bir harakatni bajarishga rag'batlantiradi yoki to‘xtatadi.  ▎2. O‘rganish Jarayoni  1. Agent va Muhit O‘rtasidagi O‘zaro Aloqa:  • Agent harakat qiladi va muhitga ta'sir etadi.  • Muhit agentga yangi holatni va mukofotni beradi.  2. Mukofotlarni Qabul Qilish:  • Agent har bir harakatdan keyin mukofot oladi. Mukofot ijobiy yoki salbiy bo'lishi mumkin.  • Ijobiy mukofot agentni ma'lum bir harakatni takrorlashga undaydi, salbiy mukofot esa ushbu harakatni bajarishdan to‘xtatadi.  3. Strategiyani Yangilash:  • Agent o‘z strategiyasini (policy) yangilash uchun mukofotlarni tahlil qiladi. Bu jarayon ko‘pincha quyidagi usullar orqali amalga oshiriladi:  • Q-learning: Agent har bir holat-harakat juftligi uchun Q qiymatini (harakatning kutilayotgan mukofoti) hisoblaydi va yangilaydi.  • Policy Gradient: Agentning strategiyasini to'g'ridan-to'g'ri optimallashtiradi, mukofotga asoslangan holda strategiyani yangilaydi.  ▎3. Q-learning Misol  Q-learning algoritmi yordamida agent qanday qilib o‘z strategiyasini rivojlantirishi mumkinligini ko'rib chiqamiz:  1. Q-Tablitsa Yaratish: Har bir holat-harakat juftligi uchun Q qiymatini saqlovchi jadval.  2. O'qish Jarayoni:  • Agent tasodifiy holatda boshlaydi va harakat qiladi.  • Har bir harakatdan so‘ng Q qiymatini yangilaydi:  Q(s, a) ← Q(s, a) + α (( r + γ max\_(a') Q(s', a') - Q(s, a) ))  Bu yerda:  • s — joriy holat,  • a — bajarilgan harakat,  • r — mukofot,  • s' — yangi holat,  • α — o‘rganish darajasi,  • γ — kelajakdagi mukofotni hisobga olish koeffitsienti.  3. Strategiyani Yangilash: Har bir holat uchun eng yuqori Q qiymatini tanlab, agent o'z strategiyasini yangilaydi.  ▎4. Natijalar  O‘rganish jarayoni davomida agent ko‘plab tajribalarni yig‘adi va mukofotlar orqali o‘z strategiyasini rivojlantiradi. Mukofotlar yordamida agent muvaffaqiyatli harakatlarni aniqlaydi va ular asosida o‘z strategiyasini optimallashtiradi.  ▎Xulosa  Reinforcement Learning algoritmlari agentlarga ma'lum bir vazifalarni bajarishni o‘rganishda juda samarali. Mukofotlar orqali agentlar o‘z strategiyalarini rivojlantirib, muhitda muvaffaqiyatli harakat qilish imkoniyatlarini oshiradilar. Bu jarayon ko‘plab real hayotdagi muammolarni hal qilishda qo‘llaniladi, masalan, o‘yinlar, robototexnika va avtomatlashtirilgan tizimlar. |
| 41. Oddiy sun'iy intellekt agenti yaratib, uni ma'lum bir muhitda maqsadga erishish uchun qidiruv algoritmlaridan (masalan, BFS yoki DFS) foydalanishga undang. Agent qanday qilib muhitda optimal yo‘lni topadi? | Oddiy sun'iy intellekt agentini yaratish va uni ma'lum bir muhitda maqsadga erishish uchun qidiruv algoritmlaridan foydalanishga undirish jarayoni quyidagi bosqichlarni o'z ichiga oladi:  ▎1. Agent va Muhitni Ta'riflash  • Agent: O‘rganayotgan subyekt (masalan, robot yoki dastur).  • Muhit: Agent harakat qiladigan maydon, masalan, kvadratlar yoki graf shaklida ifodalangan joy.  • Maqsad: Agent erishmoqchi bo‘lgan nuqta yoki holat.  ▎2. Qidiruv Algoritmlarining Tanlovi  Agentning muhitda optimal yo‘lni topishi uchun quyidagi qidiruv algoritmlaridan biri tanlanadi:  • BFS (Breadth-First Search): Har bir darajadagi barcha tugunlarni ko‘rib chiqadi va eng qisqa yo‘lni topadi.  • DFS (Depth-First Search): Har bir yo‘nalishda chuqurlik bilan harakat qiladi, lekin eng qisqa yo‘lni kafolatlamaydi.  ▎3. BFS Algoritmini Qo'llash  BFS algoritmi orqali agent qanday qilib optimal yo‘lni topishi mumkinligini ko'rib chiqamiz:  ▎3.1. Muhitni Model qilish  Muhitni graf sifatida tasavvur qilamiz, unda tugunlar (node) agentning holatlari va qirralar (edge) esa agentning harakatlari.  class Node:  def \_\_init\_\_(self, position):  self.position = position  self.children = []  # Misol uchun, 2D grid  grid = [  [0, 0, 0, 0],  [0, 1, 0, 0], # 1 - to'siq  [0, 0, 0, 0],  ]  ▎3.2. BFS Algoritmini Implementatsiya qilish  from collections import deque  def bfs(start, goal, grid):  queue = deque([start]) # Qatorni yaratamiz  visited = set() # Ko'rilgan tugunlar  visited.add(start.position)  while queue:  current\_node = queue.popleft() # Qatorning boshidan tugunni olish  if current\_node.position == goal.position:  return True # Maqsadga yetishdi  # Qo'shni tugunlarni tekshirish  for direction in [(0, 1), (1, 0), (0, -1), (-1, 0)]: # O'ng, Past, Chap, Yuqoriga harakat  new\_position = (current\_node.position[0] + direction[0], current\_node.position[1] + direction[1])  # To'siqlarni tekshirish  if (0 <= new\_position[0] < len(grid) and  0 <= new\_position[1] < len(grid[0]) and  grid[new\_position[0]][new\_position[1]] == 0 and  new\_position not in visited):    visited.add(new\_position)  queue.append(Node(new\_position)) # Yangi tugunni qo'shish  return False # Maqsadga yetisholmadi  # Boshlanish va maqsad nuqtalari  start\_node = Node((0, 0))  goal\_node = Node((2, 3))  # Algoritmni chaqirish  found = bfs(start\_node, goal\_node, grid)  print("Maqsadga yetishildi:", found)  ▎4. Optimal Yo'lni Topish  BFS algoritmi orqali agent har bir holatni ko‘rib chiqadi va eng qisqa yo‘lni topishga harakat qiladi. Har bir tugunni ko‘rib chiqish jarayonida agent to‘siqlardan o‘tmasligi va ilgari ko‘rilgan tugunlarga qaytmasligi kerak.  ▎Xulosa  Agent BFS yoki DFS kabi qidiruv algoritmlari yordamida muhitda maqsadga erishish uchun optimal yo‘lni topishi mumkin. BFS algoritmi har doim eng qisqa yo‘lni topadi, lekin bu algoritmning samaradorligi muhitning kattaligi va murakkabligiga bog'liq. DFS esa chuqurroq o'rganishga imkon beradi, lekin eng qisqa yo'l kafolatlanmaydi. Agentning muvaffaqiyati muhitning tuzilishi va tanlangan qidiruv algoritmiga bog'liq. |
| 42. Universitet ma’muriyatiga o‘quv jarayonini boshqarish bo‘yicha yordam beruvchi agent yaratish. |  |
| 43. Dars davomida talabalarning ishtirokini kuzatib boruvchi agent yarating |  |
| 44. Sun'iy intellekt agenti yaratib, ilmiy-tadqiqot loyihalarini boshqarishni avtomatlashtiring. |  |
| 45. |  |
| 46. |  |
| 47. |  |
| 48. |  |
| 49. |  |
| 50. |  |
| 51. |  |
| 52. |  |
| 60. |  |