سوال اول:

الف:

اشتباه

مقادیر کوچک k نویز را به خاطر می سپارند، و در نتیجه منجر به یک مرز تصمیم گیری غیر هموار می شود. این خطای کل را افزایش می دهد، جایی که واریانس بالا بر آن غالب است

مقادیر بزرگ k، روندهای اساسی در داده ها (ویژگی های محلی) را نادیده می گیرند، و بنابراین منجر به یک مرز تصمیم گیری صاف می شوند. این خطای کل را افزایش می‌دهد، جایی که بایاس بالا بر آن غالب است.

ب:

درست

. چون هرچه مقدار k کمتر باشد روی همین تعداد کم حساس می­شود و اگر این همسایه ها نویز باشند در نتیجه خروجی اشتباه به ما می­دهد.

سوال دوم:

الف:

عملکرد الگوریتم k-NN با افزایش تعداد ویژگی ها بدتر می شود. از این رو، تحت تأثیر نفرین ابعاد قرار می گیرد. زیرا در فضاهای با ابعاد بالا، الگوریتم k-NN با دو مشکل مواجه است:

محاسبه فاصله و یافتن نزدیکترین همسایگان در فضای با ابعاد بالا از نظر محاسباتی گرانتر می شود.

فرض ما مبنی بر قرار گرفتن نقاط مشابه از نزدیک شکسته می شود

با افزایش تعداد ویژگی ها، فاصله بین نقاط داده کمتر متمایز می شود. علاوه بر این، کل مساحتی که برای یافتن k همسایه باید پوشش دهیم افزایش می یابد.

ب:

رابطه بین الگوریتم KNN و مبادله بایاس واریانس الگوریتم KNN به مبادله بایاس واریانس بر اساس نحوه تعادل پیچیدگی مدل مرتبط است. با افزایش تعداد همسایگان (K):

سوگیری کاهش می‌یابد: مدل پیچیده‌تر می‌شود و بهتر می‌تواند با داده‌های آموزشی مطابقت داشته باشد، و تعصب نادرست را کاهش می‌دهد.

واریانس افزایش می‌یابد: این مدل با داده‌های آموزشی، از جمله نویز، افزایش شانس تطابق بیش از حد با مجموعه آموزشی و کاهش تعمیم به داده‌های جدید، نزدیک‌تر می‌شود. این باعث افزایش واریانس می شود.

بنابراین KNN یک مبادله بایاس واریانس را نشان می دهد - K پایین تر سوگیری را افزایش می دهد اما واریانس را کاهش می دهد، در حالی که K بالاتر برعکس را انجام می دهد. هدف این است که K را انتخاب کنید که خطای کل را به حداقل برساند.

بخش 2: تأثیر مقدار K بر بایاس و واریانس

یک مقدار K کوچکتر (به عنوان مثال K=1) منجر به یک مدل بسیار ساده می شود که بعید به نظر می رسد با داده های آموزشی به خوبی مطابقت داشته باشد، و در نتیجه بایاس بالا اما واریانس کم ایجاد می شود.

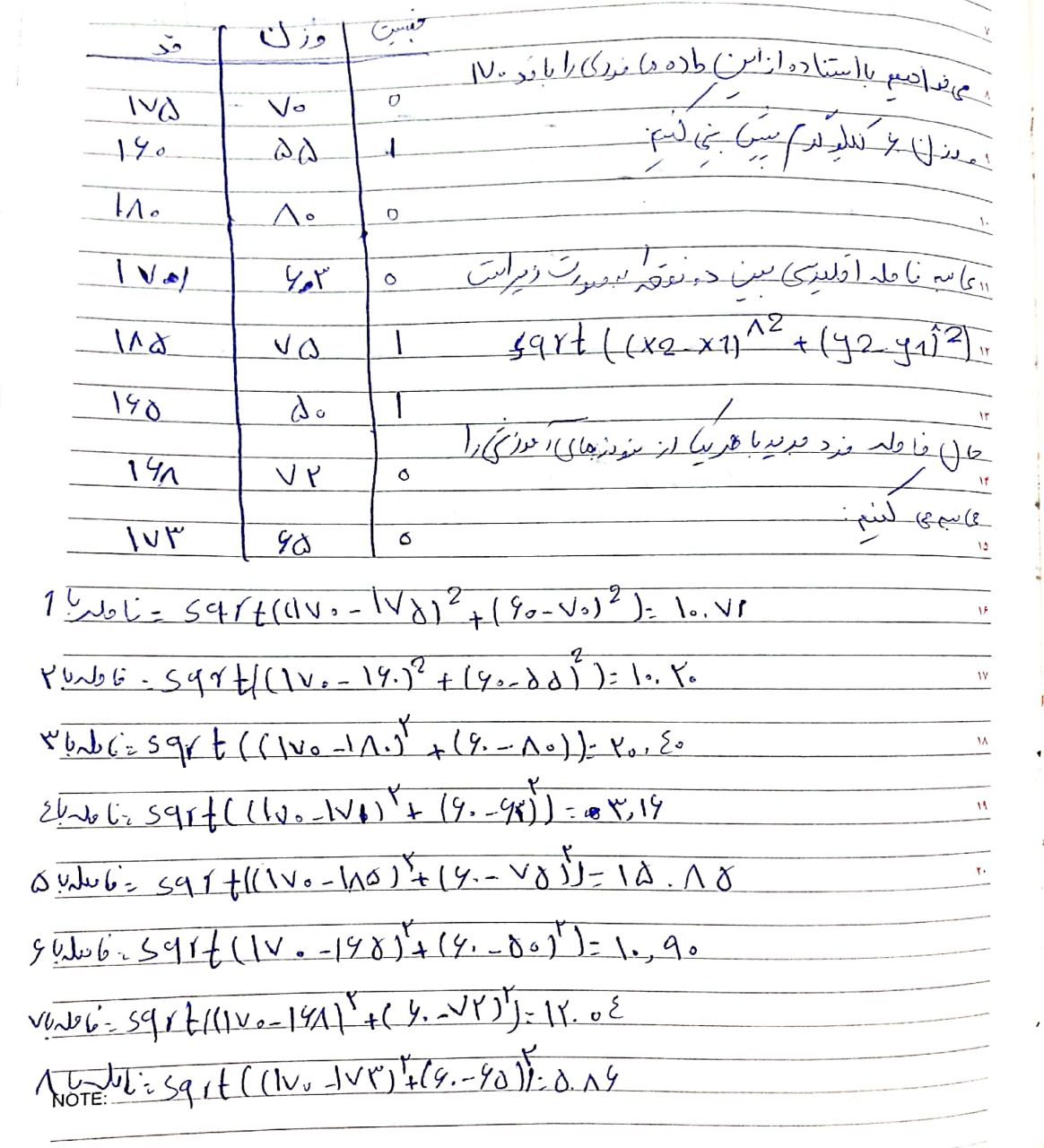
یک مقدار K بزرگتر، مدل را پیچیده‌تر می‌کند و به آن اجازه می‌دهد تا داده‌های آموزشی را بهتر تطبیق دهد، اما نویز را نیز برازش می‌کند، که منجر به بایاس کم اما واریانس بالا می‌شود.

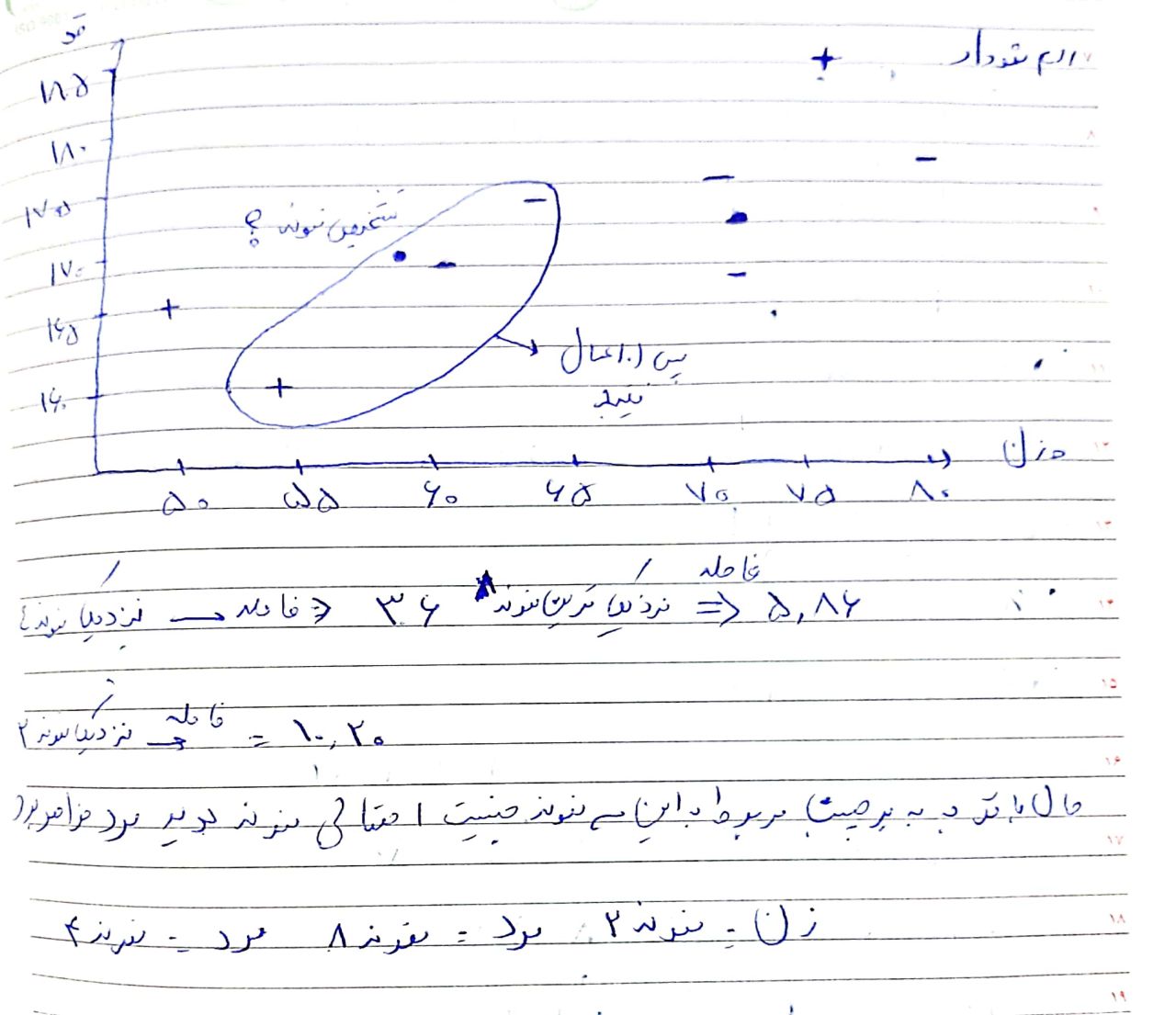
بنابراین به طور خلاصه، K کوچکتر به معنای بایاس بیشتر و واریانس کمتر است، در حالی که K بزرگتر به معنای بایاس کمتر اما واریانس بالاتر برای مدل KNN است.

ج:

پاسخ همه این معیارهای فاصله را می توان به عنوان یک متریک فاصله برای k-NN استفاده کرد.

د:





سوال سوم:

1NN:

ما هنوز یک نقطه داده را نگه می داریم، اما به سه نزدیک باقیمانده اجازه می دهیم اکثریت را در مورد نحوه طبقه بندی نقطه باقیمانده رای دهند. خوشه سمت راست کم و بیش یکسان است.

در خوشه سمت چپ، اگر هر یک از X ها را در اطراف لبه ها نگه داریم، نزدیک ترین نقاط باقی مانده یک X ، یک – هستند، بنابراین اکثریت رای یک X است، و ما درست می گوییم. اگر - را در وسط نگه داریم، نزدیکترین درخت باقی مانده همه X است، بنابراین ما به عنوان X طبقه بندی می کنیم، که نادرست است.

در مجموع از هر ده بار یکی اشتباه کردیم.یعنی 10/1

3NN:

هر گاه یکی از نقاط خوشه سمت راست را بیرون بیاوریم، به درستی طبقه بندی می شود: یک - را نگه می داریم، نزدیکترین نقطه به نقطه باز شده نیز - است.

در خوشه سمت چپ، ما همیشه نقطه باز شده را به اشتباه طبقه بندی می کنیم. اگر نقطه بیرون نگه داشته شده یکی از X ها در مربع باشد، نقطه بسته باقی مانده یک - است، بنابراین طبقه بندی ما نادرست است. اگر نقطه بیرون نگه داشته شده - در وسط باشد، نزدیکترین نقطه با یک + است، بنابراین در این مورد نیز نادرست هستیم.

در مجموع، ما 5 مورد از 10 احتمال را به اشتباه طبقه بندی کردیم. یعنی 10/5

سوال چهارم:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| فاصله چیبیشف با 9 | فاصله منهتن با 9 | class | X2 | X1 | ردیف |
| 10 | 15 |  | 5 | 5- | 1 |
| 10 | 10 |  | 0 | 5- | 2 |
| 10 | 15 |  | 5- | 5- | 3 |
| 5 | 5 |  | 0 | 0 | 4 |
| 5 | 10 |  | 5- | 0 | 5 |
| 5 | 10 |  | 5 | 0 | 6 |
| 2 | 2 |  | 2- | 5 | 7 |
| 3 | 8 |  | 5 | 8 | 8 |
| 1-NN =   3-NN = | 1-NN =   3-NN = |  | 0 | 5 | 9 |

سوال پنجم:

الف) k = 1 ب) k = 3 ج) k = 7

سوال ششم:

الف:

* توابع پایه رادیال مانند توابع گوسی هستند که بر اساس فاصله نمونه‌ها از مرکز تابع تعریف می‌شوند.
* این توابع در نقاطی که فاصله کمتری با مرکز تابع دارند، مقدار بیشتری را برمی‌گردانند.
* توابع RBF غیرخطی هستند و برای طبقه‌بندی غیرخطی مناسب‌تر از شبکه‌های عصبی پرسپترون هستند.
* در RBF، هر نمونه توسط یک تابع پایه نمایش داده می‌شود و وزن‌های بین توابع پایه و لایه خروجی تعیین می‌کند که نمونه به چه کلاسی تعلق دارد.
* تعداد پارامترهای RBF کمتر از شبکه‌های عصبی معمولی است که آموزش آن‌ها سرعت بیشتری دارد.

ب:

استفاده از تقریب در تابع به این معنی است که با استفاده از روش‌ها و الگوریتم‌های مختلف، سعی می‌شود تابع مورد نظر را به صورت تقریبی تقریب زد. این کار به دست آوردن یک تقریب نزدیک به مقدار واقعی تابع در نقاط مختلف فضا کمک می‌کند.

برای تقریب تابع، می‌توان از روش‌های مختلفی استفاده کرد. برخی از روش‌های معروف تقریب تابع عبارتند از:

* روش تقسیم و حل: در این روش، با تقسیم بازه مورد نظر به بخش‌های کوچکتر و استفاده از روش‌های تقریبی در هر بخش، تقریبی از تابع به دست می‌آید.
* روش تقریب با استفاده از توابع پایه: در این روش، تابع مورد نظر با استفاده از ترکیب خطی از توابع پایه مشخصی تقریب زده می‌شود. توابع پایه می‌توانند توابع ساده‌ای مانند توابع چندجمله‌ای یا توابع گوسی باشند.
* روش تقریب با استفاده از روش‌های آماری: در این روش، با استفاده از داده‌های موجود و روش‌های آماری مانند رگرسیون، تابع مورد نظر تقریب زده می‌شود.

ج:

توابع پایه رادیال و شبکه عصبی هر دو روش‌هایی هستند که برای تقریب توابع پیچیده مورد استفاده قرار می‌گیرند. با این حال، روش آموزش و عملکرد آن‌ها تفاوت‌هایی دارد.

در شبکه عصبی، شبکه از لایه‌های مختلف تشکیل شده است که هر لایه شامل یک تعداد نورون‌ها است. اطلاعات از لایه ورودی به لایه‌های میانی و سپس به لایه خروجی منتقل می‌شود. آموزش شبکه عصبی شامل مراحل تعیین وزن‌ها و بایاس‌ها و بهینه‌سازی آن‌ها با استفاده از الگوریتم‌های مختلف مانند الگوریتم پس‌انتشار خطا است.

در مقابل، توابع پایه رادیال توسط یک مرکز و وزن‌ها تعریف می‌شوند. هر نمونه توسط یک تابع پایه نمایش داده می‌شود و وزن‌ها بین توابع پایه و لایه خروجی تعیین می‌شوند. آموزش توابع پایه رادیال شامل تعیین مراکز و وزن‌ها و بهینه‌سازی آن‌ها با استفاده از روش‌های مختلف مانند روش کمترین مربعات است.

به طور خلاصه، هر دو توابع پایه رادیال و شبکه عصبی برای تقریب توابع پیچیده قابل استفاده هستند، اما روش آموزش و عملکرد آن‌ها متفاوت است.

نحوه آموزش توابع پایه رادیال به صورت زیر است:

1. انتخاب مراکز: در ابتدا، باید مراکز توابع پایه رادیال را انتخاب کنیم. این مراکز معمولاً بر اساس نمونه‌های داده‌های آموزشی انتخاب می‌شوند. می‌توان از روش‌های مختلفی مانند روش k-means برای انتخاب مراکز استفاده کرد.
2. تعیین وزن‌ها: بعد از انتخاب مراکز، باید وزن‌ها بین توابع پایه و لایه خروجی تعیین شوند. این وزن‌ها معمولاً با استفاده از روش‌های بهینه‌سازی مانند روش کمترین مربعات تعیین می‌شوند.
3. آموزش شبکه: پس از تعیین مراکز و وزن‌ها، شبکه برای آموزش آماده است. در این مرحله، از داده‌های آموزشی استفاده می‌شود تا شبکه آموزش داده شود. این شامل ارائه نمونه‌های آموزشی به شبکه و به‌روزرسانی وزن‌ها بر اساس خطا است.
4. ارزیابی و تست: پس از آموزش شبکه، باید عملکرد آن را ارزیابی کنید. این شامل استفاده از داده‌های تست برای ارزیابی دقت و عملکرد شبکه است.

د:

1. روش تقسیم و حل: یکی از روش‌های معمول برای انتخاب گره‌های مخفی، استفاده از روش تقسیم و حل است. در این روش، با تقسیم بازه مورد نظر به بخش‌های کوچکتر، مراکز توابع پایه رادیال را انتخاب می‌کنیم. این روش می‌تواند به صورت خودکار یا با استفاده از الگوریتم‌های بهینه‌سازی مانند روش k-means انجام شود.
2. روش انتخاب تصادفی: روش دیگری برای انتخاب گره‌های مخفی، استفاده از روش انتخاب تصادفی است. در این روش، مراکز توابع پایه رادیال به صورت تصادفی از دامنه مورد نظر انتخاب می‌شوند.
3. روش انتخاب مبتنی بر داده: روش دیگری که ممکن است استفاده شود، استفاده از داده‌های آموزشی برای انتخاب گره‌های مخفی است. در این روش، می‌توان مراکز توابع پایه رادیال را بر اساس نمونه‌های داده‌های آموزشی انتخاب کرد.

سوال هفتم: **درخت تصمیم**: در صورت استفاده از درخت تصمیم، می‌توانیم برچسب‌های مثبت و منفی را معکوس کنیم. به عنوان مثال، برچسب‌های منفی را به عنوان برچسب‌های مثبت و برچسب‌های مثبت را به عنوان برچسب‌های منفی در نظر بگیریم.

**شبکه عصبی**: در شبکه عصبی، می‌توانیم تابع فعال‌سازی خروجی را تغییر دهیم. به عنوان مثال، اگر از تابع سیگموید استفاده می‌کنیم، می‌توانیم از تابع سیگموید معکوس (تابع لگاریتمیت) استفاده کنیم.

**نزدیکترین همسایه**: در الگوریتم نزدیکترین همسایه، می‌توانیم برچسب‌های مثبت و منفی را معکوس کنیم. به عنوان مثال، برچسب‌های منفی را به عنوان برچسب‌های مثبت و برچسب‌های مثبت را به عنوان برچسب‌های منفی در نظر بگیریم.

سوال هشتم:

**رویکرد درخت تصمیم (Decision Tree):**

* نقاط قوت:
  + قابلیت فهم و تفسیر آسان: درخت تصمیم به صورت گرافیکی قابل فهم است و می‌تواند قوانین قابل فهمی را برای تصمیم‌گیری ارائه دهد.
  + قابلیت کاربرد در داده‌های کم و بزرگ: درخت تصمیم به خوبی با داده‌های کم و بزرگ کار می‌کند و می‌تواند بازدهی مناسبی داشته باشد.
  + قابلیت مدل‌سازی ویژگی‌های غیرخطی: درخت تصمیم قادر است به خوبی با ویژگی‌های غیرخطی مانند تعامل و تأثیر متقابل بین ویژگی‌ها سازگاری پیدا کند.
* نقاط ضعف:
  + تمایل به برازش بیش از حد (Overfitting): درخت تصمیم ممکن است در صورتی که عمق آن زیاد شود، به داده‌های آموزشی بسیار خوب برازش شود اما در مقابل داده‌های جدید ناشناخته نتایج نامطلوبی ارائه دهد.
  + حساسیت به تغییرات کوچک در داده‌ها: درخت تصمیم ممکن است با تغییرات کوچک در داده‌ها تغییرات بزرگی در ساختار درخت داشته باشد.

**الگوریتم نزدیکترین همسایه K-Nearest Neighbors (KNN):**

* نقاط قوت:
  + سادگی و قابلیت پیاده‌سازی: KNN یک الگوریتم ساده و قابل فهم است و برای پیاده‌سازی نیاز به تنظیم پارامترهای پیچیده ندارد.
  + عملکرد خوب در مجموعه داده‌های بزرگ: KNN به خوبی با مجموعه داده‌های بزرگ کار می‌کند و می‌تواند نتایج قابل قبولی را در زمان معقولی ارائه دهد.
  + قابلیت مدل‌سازی ویژگی‌های غیرخطی: KNN قادر است به خوبی با ویژگی‌های غیرخطی مانند تعامل و تأثیر متقابل بین ویژگی‌ها سازگاری پیدا کند.
* نقاط ضعف:
  + نیاز به مقداردهی مناسب پارامتر K: انتخاب درست مقدار K (تعداد همسایگان نزدیک) در KNN بسیار مهم است و ممکن است نتایج را تحت تأثیر قرار دهد.
  + حساسیت به داده‌های پرت: KNN حساسیت بالایی به داده‌های پرت دارد و ممکن است نتایج نامطلوبی در صورت وجود داده‌های پرت ارائه دهد.

برای مثال، در صورتی که داده‌ها دارای تعداد بزرگی ویژگی باشند و تعاملات پیچیده‌ای بین ویژگی‌ها وجود داشته باشد، رویکرد درخت تصمیم ممکن است بهترین عملکرد را داشته باشد. از سوی دیگر، اگر داده‌ها دارای تعداد کمی ویژگی باشند و تأثیر ویژگی‌ها به صورت خطی باشد، الگوریتم نزدیکترین همسایه KNN ممکن است بهترین عملکرد را داشته باشد.

توجه داشته باشید که این مثال‌ها فقط برای توضیح نقاط قوت و ضعف هر روش است و در هر مورد خاص، نتیجه‌گیری باید بر اساس آزمون و ارزیابی دقیق‌تر انجام شود.

در زیر نمونه‌هایی از سناریوهایی که در آن یک الگوریتم ممکن است از دیگری بهتر عمل کند را ارائه می‌دهیم:

1. **حجم داده بزرگ**: در صورتی که مجموعه داده بسیار بزرگ باشد، الگوریتم نزدیکترین همسایه KNN ممکن است بهتر عمل کند. زیرا در این حالت، محاسبه فاصله بین نقاط و تشخیص همسایگان نزدیک زمان بیشتری می‌برد و الگوریتم KNN به خوبی با داده‌های بزرگ کار می‌کند.
2. **ویژگی‌های غیرخطی**: در صورتی که ویژگی‌ها تعاملات پیچیده و غیرخطی داشته باشند، رویکرد درخت تصمیم ممکن است بهتر عمل کند. زیرا درخت تصمیم قادر است به خوبی با ویژگی‌های غیرخطی سازگاری پیدا کند و قوانین قابل فهمی را برای تصمیم‌گیری ارائه دهد.
3. **تعداد ویژگی‌ها کم**: در صورتی که تعداد ویژگی‌ها کم باشد و تأثیر ویژگی‌ها به صورت خطی باشد، الگوریتم نزدیکترین همسایه KNN ممکن است بهتر عمل کند. زیرا در این حالت، محاسبه فاصله بین نقاط و تشخیص همسایگان نزدیک به سرعت انجام می‌شود و الگوریتم KNN ساده و قابل فهمی است.

سوال نهم:

الف:

1. **مفهوم سازگاری در Dataset Condensed و عیب آن:**
   * **سازگاری در Dataset Condensed:** مفهوم سازگاری به تطابق یا هماهنگ بین داده‌های موجود در مجموعه داده کاهش‌یافته (Condensed Dataset) و داده‌های موجود در مجموعه داده کامل (Full Dataset) اشاره دارد. به عبارت دیگر، اگر از یک مدل مانند K-NN بر روی مجموعه داده کاهش‌یافته استفاده شود، نتایج پیش‌بینی باید با نتایج مدل بر روی مجموعه داده کامل سازگار باشند.
   * **عیب‌ها:**
     + **زمان و محاسبات:** ایجاد مجموعه داده کاهش‌یافته ممکن است به محاسبات زمانی بالا و هزینه‌ی محاسباتی افزوده شود، زیرا برای اطمینان از سازگاری، نیاز به مقایسه و بررسی دقیق داده‌ها وجود دارد.
     + **احتمال از دست رفتن اطلاعات:** درصورتی که مجموعه داده کاهش‌یافته به‌طور ناقصی سازگار شود، ممکن است اطلاعات مهمی از مجموعه داده اصلی از دست رفته باشند.
2. **تفاوت با سازگاری مجموعه داده آموزش:**
   * **سازگاری مجموعه داده آموزش:** وقتی از سازگاری در مجموعه داده آموزش صحبت می‌شود، اشاره به تطابق بین داده‌های موجود در مجموعه داده مورد استفاده برای آموزش یک مدل (مانند مدل K-NN) و داده‌های واقعی در محیط آموزش دارد. این سازگاری نشان‌دهنده‌ی اطمینان از اینکه داده‌های موجود در مرحله آموزش نمایانگر واقعیت داده‌های محیط زندگی مورد نظر هستند.
   * **تفاوت:** در مقایسه با سازگاری در Dataset Condensed، سازگاری مجموعه داده آموزش مرتبط با مطمئن شدن از اعتبار و اثربخشی مدل در فاز آموزش است، در حالی که سازگاری در Dataset Condensed بیشتر به تطابق نتایج مدل در مرحله پیش‌بینی (deployment) بر روی داده‌های مقایسه شده می‌پردازد.

ب:

الگوریتم نزدیکترین همسایه متراکم (Condensed Nearest Neighbor - CNN) یک الگوریتم هیوریستیک برای ساخت یک مجموعه داده متراکم است. هدف اصلی این الگوریتم از نظر محاسباتی، کاهش حجم مجموعه داده بدون افت زیاد در عملکرد مدل نزدیکترین همسایه (k-NN) است. الگوریتم به صورت مراحل زیر عمل می‌کند:

1. **انتخاب نقاط اولیه:** نقاط اولیه به صورت تصادفی از مجموعه داده انتخاب می‌شوند. این نقاط نماینده اولیه مجموعه داده کاهش‌یافته خواهند بود.
2. **پیش‌بینی با مدل:** از یک مدل نزدیکترین همسایه (مانند K-NN) بر روی نقاط انتخاب شده استفاده می‌شود تا نقاطی که به نظر مدل اهمیتی ندارند حذف شوند.
3. **افزودن نقاط مهم:** برخی از نقاطی که مدل اعتقاد دارد مهم هستند (یعنی پیش‌بینی مدل برای آنها متفاوت از مقدار واقعی است) به مجموعه داده متراکم اضافه می‌شوند.
4. **تکرار مراحل 3 و 4:** مراحل 3 و 4 به صورت تکراری اجرا می‌شوند تا اطمینان حاصل شود که مدل به خوبی متراکم شده و قابلیت پیش‌بینی حفظ شده است.
5. **اتمام الگوریتم:** الگوریتم در زمانی مشخص خاتمه می‌یابد (به طور معمول، تعداد تکرارها محدود می‌شود).

الگوریتم نزدیکترین همسایه متراکم به صورت ساده و قابل فهم است و معمولاً باعث افزایش سرعت پردازش مدل در مراحل پیش‌بینی می‌شود.

**ویژگی‌ها و مزایا:**

* **کاهش حجم داده:** الگوریتم CNN با اضافه کردن تنها نقاط لازم به مجموعه داده، به کاهش حجم داده کمک می‌کند.
* **حفظ سازگاری:** مجموعه داده متراکم حاصل، برای نقاط آموزش موجود در مجموعه داده اصلی، نتایج k-NN یکسان با مجموعه داده اصلی داشته و سازگار با آن است.

سوال دهم:

مختصات ستاره به صورت (0.7، 0.6) مشخص است. تعداد نقاط با کلاس مشکی (کلاس 1) برابر با 59 و تعداد نقاط با کلاس آبی (کلاس 2) برابر با 61 است. حالا، پنج همسایه نزدیکترین به ستاره که به کلاس آبی تعلق دارند را انتخاب می‌کنیم و یک دایره با شعاع برابر با فاصله بین ستاره و دورترین همسایه‌اش (در اینجا ممکن است (0.6، 0.5) یا (0.6، 0.7) باشد) رسم می‌کنیم.

مختصات همسایه‌های انتخاب شده به ترتیب (0.8، 0.6)، (0.7، 0.8)، (0.6، 0.5)، (0.6، 0.6) و (0.6، 0.7) هستند. فرض می‌کنیم فاصله بین ستاره و دورترین همسایه، به طور یکسان از ستاره باشد. بنابراین، شعاع دایره برابر با:

r2 = = 0.1 = r

حالا، حجم (یا مساحت در دو بعد) دایره که پنج همسایه‌ای از کلاس آبی در آن قرار گرفته‌اند، برابر با:

V2 = πr2

سپس، پنج همسایه نزدیکترین به ستاره که به کلاس مشکی تعلق دارند را انتخاب می‌کنیم و یک دایره با شعاع برابر با فاصله بین ستاره و دورترین همسایه‌اش رسم می‌کنیم. مختصات همسایه‌های انتخاب شده به ترتیب (0.7، 0.5)، (0.8، 0.7)، (0.8، 0.5)، (0.9، 0.6) و (0.9، 0.8) هستند. فرض می‌کنیم دورترین همسایه (0.9، 0.8) باشد. بنابراین، شعاع دایره برابر با:

r1 = = 0.2

با توجه به اینکه r1 = 2 r

، حجم دایره که پنج همسایه‌ای از کلاس مشکی در آن قرار گرفته‌اند برابر با:

V1 = 4πr2

حالا با توجه به فرمول تخمین چگالی با *k* نزدیکترین همسایه، احتمال متعلق بودن ستاره به کلاس مشکی (کلاس 1) برابر با:

p(x) =

احتمال متعلق بودن نقطه ستاره به کلاس مشکی(کلاس 1) برابر است با :

p(x) =

احتمال متعلق بودن نقطه ستاره به کلاس آبی(کلاس 2) برابر است با :

p(x) =

بنابراین نقطه ستاره متعلق به کلاس آبی(کلاس 2) است.

سوال یازدهم:

یک مثال عددی ساده برای استفاده از الگوریتم وزن‌دار Locally Weighted Regression (LWR) در پیش‌بینی قیمت خانه را بررسی میکنیم. در اینجا، فرض میکنیم مجموعه داده دو ویژگی دارد: اندازه خانه و فاصله از مرکز شهر. همچنین، می‌خواهیم قیمت خانه نوساز را پیش‌بینی کنیم.

**مجموعه داده:**

داده ها به صورت زیر است:

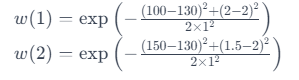
|  |  |  |
| --- | --- | --- |
| قیمت خانه(میلیون) | فاصله از مرکز شهر(کیلومتر) | اندازه خانه(متر مربع) |
| 300 | 2 | 100 |
| 500 | 1.5 | 150 |
| 400 | 3 | 120 |
| 600 | 2.2 | 180 |

حالا از الگوریتم وزن‌دار LWR برای پیش‌بینی قیمت خانه برای یک مورد نقطه‌ای (مثلاً خانه ای با اندازه 130 متر مربع و فاصله از مرکز شهر 2 کیلومتر) استفاده میکنیم.

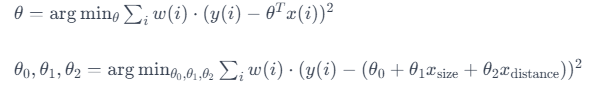
1. **تعیین نقطه مورد نظر (Query Point):** اندازه خانه = 130 متر مربع فاصله از مرکز شهر = 2 کیلومتر
2. **محاسبه وزن‌ها (Weights):** استفاده از تابع وزن‌دهی:



مثلاً با فرض 1*τ*:



**فیت مدل LWR:** محاسبه *θ* با مینیمم کردن مجموع وزن‌دار مربعات خطا:



1. **پیش‌بینی قیمت:** با استفاده از مدل به دست آمده، پیش‌بینی قیمت خانه: Predicted\_Price=*θTx*

این مثال نشان می‌دهد چگونه با استفاده از الگوریتم وزن‌دار LWR، مدل به داده‌های نزدیک به نقطه مورد نظر بیشتر توجه می‌کند و میانگین خطای پیش‌بینی ممکن است بهبود یابد.