

پاسخ سوال ۱:

(الف) در تخصیص یک کلاس به یک داده تست بر اساس الگوریتم k -NN، زمانیکه تمام k همسایه از یک کلاس باشند، داده تست در همان کلاس قرار خواهد گرفت اما زمانیکه برخی از k همسایه، در یک کلاس و برخی دیگر در کلاس یا کلاسهای دیگری باشند، در اینجا باید با استفاده از روشهایی، بین کلاسهای مختلف، voting انجام بگیرد تا مشخص شود که داده تست را باید در کدام کلاس قرار دهیم.^۱

(ب) از همبستگی spearman زمانی استفاده می شود که ویژگی ها، توزیع نرمالی ندارند.^۲

(پ) یکی از ضعف های اصلی معیار لایبلر - کول بک این است که معیار نیست! زیرا شرط تقارن را برآورده نمی کند. یک ضعف بزرگتر آن، این است که این معیار به دلیل وجود مرزهای bin ها مستعد خطاست. فاصله بین یک تصویر و یک نسخه کمی تیره تر از خودش می تواند زیاد باشد، اگر پیکسل ها در یک bin مجاور بیفتند، زیرا در این معیار، مجاورت bin ها در نظر گرفته نمی شود.^۳

(ت) از آنجایی که تمام کارها در زمان اجرا انجام می شود، در صورتی که مجموعه آموزشی بزرگ باشد، k -NN می تواند عملکرد زمان اجراء ضعیفی داشته باشد. همچنین k -NN به ویژگی های نامربوط یا اضافی بسیار حساس است اما این مشکل را می توان با انتخاب دقیق ویژگی یا وزن دهی ویژگی بهبود بخشید. در نهایت، در کارهای طبقه بندی بسیار دشوار، k -NN ممکن است نسبت به تکنیک های دیگر مانند ماشین های بردار پشتیبان یا شبکه های عصبی ضعیف تر عمل کند.^۴

(ث) از هر دو روش برای کاهش پیچیدگی محاسباتی k -NN استفاده می شود. در اینجا ما روش Kd-Tree را توضیح می دهیم، سپس برخی مزایا و معایب هر دو را بیان می کنیم. روش Kd-Tree فضای مجموعه داده ها را بصورت یک درخت باینری بر اساس انتخاب پشت سرهم ویژگی ها، بصورت عمود بر محورها، پارتیشن بندی می کند. بهتر است که ویژگی ها به ترتیبی انتخاب شوند که آنهایی که بیشترین واریانس را دارند، در سطح بالاتری از درخت قرار بگیرند.

^۱ ر.ک. صفحه ۳ مقاله

^۲ ر.ک. صفحه ۵ مقاله

^۳ ر.ک. صفحه ۶ مقاله

^۴ ر.ک. صفحه ۲۲ مقاله

این باعث می‌شود که فضای جستجو کوچکتر شده، در نتیجه زمان جستجو نیز کاهش یابد؛ هر چند که اگر تعداد همسایه‌ها یعنی K در هنگام محاسبه زیاد باشد، سرعت آن از جستجوی **brute force** هم بیشتر خواهد شد. مشکل دیگر این روش نفرین ابعاد است یعنی اگر تعداد ویژگی‌ها زیاد باشد، پیچیدگی محاسباتی این روش از **KNN** عادی بهتر نخواهد بود.

در حالیکه روش فوق، داده‌ها را بر اساس ویژگی‌شان، پارتیشن‌بندی می‌کند، روش **Ball-Tree** یک روش بر اساس متریک است که داده‌ها را بر اساس فواصل جفت نمونه‌های آن، دسته‌بندی می‌کند. در مقایسه با **Kd-Tree**، روش **Ball-Tree** پتانسیل عملکرد بهتری برای داده‌های با ابعاد بالا دارد، به عنوان مثال، در تجزیه و تحلیل تصویر.^۵

پاسخ سوال ۲:

الف و ب و پ و ت و ث

در این قسمت‌ها، بردار ویژگی را صرفاً ۳۲ حرف الفبای فارسی در نظر می‌گیریم. در قسمت (ج)، برخی حروف عربی را نیز به بردار ویژگی اضافه می‌کنیم و مجدداً محاسبات را انجام خواهیم داد. محاسبات بوسیله زبان برنامه‌نویسی پایتون و در محیط **Google Colab** انجام شده است. در اینجا صرفاً نتایج محاسبات را در جدول آورده‌ایم. کدها نیز از این لینک^۶ در **GitHub** قابل دسترسی هستند.

Prediction Error (%) K = 5			Prediction Error (%) K = 3			Prediction Error (%) K = 1			ویژگی معیار فاصله
Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	
0	10	20	10	10	20	20	10	20	BoW دودویی
0	0	0	0	0	10	0	0	0	BoW وزندار
0	0	0	0	0	0	0	0	0	BoW نرمال شده طول
0	0	0	0	0	0	0	0	0	BoW نرمال شده نمره-زد

^۵ ر.ک صفحه ۱۰ و ۱۱ مقاله

^۶ <https://github.com/my7amin/MLPR/tree/master/Exercise-1>

(ج)

در این قسمت می‌خواهیم که چند حرف عربی نیز به بردار ویژگی اضافه کنیم که در اینصورت طول بردار ویژگی ۴۰ حرف خواهد شد و مجدداً خطا را با روشهای قبلی محاسبه کنیم.

Prediction Error (%) K = 5			Prediction Error (%) K = 3			Prediction Error (%) K = 1			ویژگی معیار فاصله
Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	
0	0	0	0	0	0	0	0	0	BoW دودویی
0	0	0	0	0	0	0	0	0	BoW وزندار
0	0	0	0	0	0	0	0	0	BoW نرمال شده طول
0	0	0	0	0	0	0	0	0	BoW نرمال شده نمره-زد

در اینجا می‌بینیم که تمام مقادیر خطاها صفر شد.

بار دیگر می‌خواهیم، صرفاً از برخی حروف فارسی و برخی حروف عربی استفاده کنیم و با اینکار طول بردار ویژگی را به صرفاً ۱۳ حرف کم کنیم و مجدداً محاسبات را انجام داده و مقادیر خطاها را در جدول وارد کنیم.

Prediction Error (%) K = 5			Prediction Error (%) K = 3			Prediction Error (%) K = 1			ویژگی معیار فاصله
Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	Correlation	Cosine	Euclidean	
0	0	0	0	0	0	0	0	0	BoW دودویی
0	0	0	0	0	0	0	0	0	BoW وزندار
0	0	0	0	0	0	0	0	0	BoW نرمال شده طول
0	0	0	0	0	0	0	0	0	BoW نرمال شده نمره-زد

این بار نیز تمام مقادیر خطاها صفر شد با این تفاوت که طول بردار ویژگی از ۴۰ حرف به ۱۳ حرف کاهش یافت. نتیجه می‌گیریم که با انتخاب هوشمندانه بردار ویژگی می‌توانیم سربار محاسباتی را کاهش و سرعت اجرا را بالاتر ببریم.