حل تمرین درس یادگیری ماشین

سری ۲

نام و نام خانوادگی دانشجو:

همایون حیدرزاده (۹۵۱۳۱۰۷۰)

نام استاد: دکتر ناظرفرد

آبان ۹۶

بخش اول

۲-

الگوریتم KNN الگوریتمی است که برای دسته بندی داده مورد نظر فاصله آن را با نزدیکترین همسایگان محاسبه کرده و بر اساس اکثریت کلاس دادههای همسایه، داده مورد نظر را دسته بندی میکند حال اگر مقدار k کوچک باشد هر داده فقط با یک همسایه مقایسه میشود و اگر نقطه ای نزدیک نقطه تست باشد ممکن است نزدیکترین همسایه آن دادهای دیگر از کلاس دیگر باشد و به عبارتی با اندکی جابجایی نقطه تست تصمیم الگویتم عوض میشود این که به معنای واریانس بالاست همانگونه که در رگرسیون در حالتی که اگر مدل تمام دادههای اموزشی را برازش کند به این معنی است که با اندکی جابجایی نقطه مورد آزمایش، به میزان زیادی مقدار پیشبینی تغییر میکند و می‌توان گفت الگوریتم دارای ناپایداری زیاد است.

در مورد بایاس فرض کنید مقدار k زیاد باشد در نتیجه هر نقطه مورد آزمایش با تعداد بیشتری نقاط آموزشی مورد مقایسه قرار میگیرد و در نتیجه درصد اطمینان برای دسته‌بندی بیشتر است حال اگر نقطهای دیگر نزدیک نقطه آزمون باشد و بخواهیم آن را دسته‌بندی کنیم میتوان گفت چون مقدار k زیاد است تقریبا با همان نقاطی مقایسه می‌شود که نقطه آزمون نزدیک آن مقایسه می‌شود بنابراین تصمیم الگوریتم به سختی عوض خواهد شد که این به معنی بایاس بالا است.

۳-

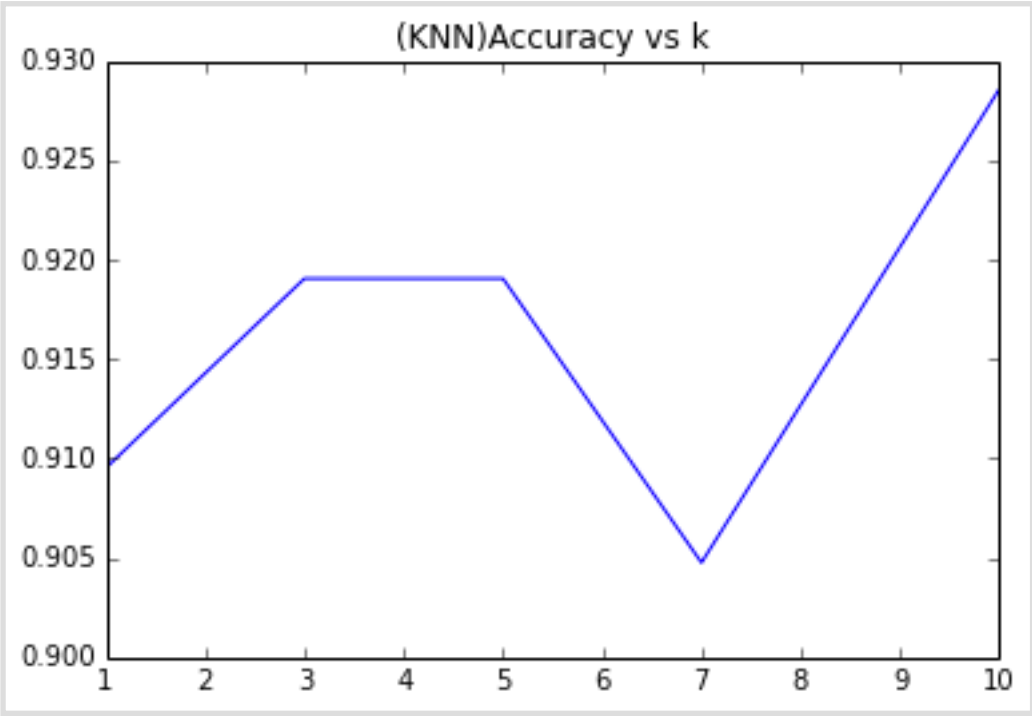
الگوریتم پارامتری دو ویژگی دارد. اول اینکه توزیع دادهها از طریق مدل ارائه شده توسط الگوریتم مشخص میشود و دوم اینکه با افزایش تعداد دادهها، پارامترها تغییر نمیکنند اما در مورد KNN باید بگوییم اگر تعداد دادهها زیاد شوند مقدار K نیز برای عملکر خوب این الگوریتم باید زیاد شود و در غیر این صورت curse of dimensinality رخ میدهد و الگوریتم KNN هیچ پیشفرضی در مورد توزیع دادهها ارائه نمیدهد چون دارای فاز یادگیری نمیباشد و فقط با استفاده از دادههای آموزشی و فاصله آنها تا دادههای تست، عمل دسته‌بندی را انجام می‌دهد و به نوعی فقط دارای مرحله تست میباشد.

۴-

[فایل پیاده‌سازی این قسمت: pr4_knn.ipynb]

آ)

نمودار دقت دسته‌بندی داده آموزشی بر اساس K در جدول زیر آمده است. برای انجام آزمایش ابتدا داده‌ها نرمال شدند.



همچنین مقادیر دقت مربوط به هر یک از kها در جدول زیر نیز آورده شده است. برای انجام آزمایش از ۱۰-fold cross validation استفاده شده است. با توجه به تغییرات دقت می‌توان گفت که این مجموعه داده دارای توزیع یکنواخت در تمامی جهات است. در نتیجه دقت آن برای k های متفاوت زیاد بایاس نمی‌شود.

برای زمانی که k یک می‌شود دقتی کمی پایین‌تر است، زیرا احتمالا در مرزهای بین داده‌ها این مقدار k خیلی مناسب نخواهد بود. همچنین تغییرات دقت برای k=۷ و k=۱۰ احتمالا به خاطر الگوهای ناهمگون موجود در داده‌ها بوده است.

K	Accuracy
1	0.90952381
3	0.91904762
5	0.91904762
7	0.9047619
10	0.92857143

ب)

این آزمایش با شرایط قسمت قبل انجام شد، با این تفاوت که cross validation روی نوع فاصله مورد استفاده صورت گرفت. نتایج به دست آمده به صورت زیر بود:

فاصله اقلیدسی و منهتن به ترتیب فواصل مینکوفسکی‌ای با p=۲ و p=۱ هستند. بنابرین مشاهده می‌شود که برای فاصله مینکوفسکی تغییرات دقت جزئی است. همچنین دقت فاصله کسینوسی اندکی کمتر است که باعث می‌شود معیار مناسبی برای این مجموعه داده نباشد.

Distance	Accuracy
euclidean	0.919047619047619
manhattan	0.919047619047619
Cosine	0.9047619047619048
Minkowski(p=4.0)	0.919047619047619
Minkowski(p=0.5)	0.9095238095238095

۵-

آ)

اولا KNN الگوریتمی ساده است و تقریبا برای هر نوع داده ساختاری مناسب می‌باشد. عدم نیاز به مدل کردن نمومه‌ها این الگوریتم را بسیار کارآمد می‌کند. از معایب این الگوریتم می‌توان به سرعت پایین و پیچیدگی محاسباتی بالا اشاره کرد، زیرا به ازای هر داده آزمایشی ورودی نیاز به محاسبات بر روی تمام داده‌های موجود است.

یکی دیگر از معایب به دلیل استفاده از فاصله معمول اقلیدسی است. محاسبه فاصله اقلیدسی پیچیدگی محاسباتی بالایی دارد و دقت الگوریتم ناپایدار می‌باشد. همچنین فاصله اقلیدوسی نمی‌تواند برای داده‌های همبسته معیار فاصله مناسبی ارائه دهد.

ب)

برای بهبود KNN روش‌های پیشنهاد شده است، از جمله: وینبرگر روشی بر پایه فاصله ماهالانوبیس به عنوان معیاری برای اندازه‌گیری پیشنهاد کرد (LMNN)، مین و همکارانش از یک نگاشت غیرخطی ویژگی بر پایه شبکه‌های عصبی عمیق برای بهبود KNN استفاده کردند (Dnet-kNN)، ینگ و همکارانش روش WDkNN را پیشنهاد دادند که بر پایه شباهت وزن‌دار k نزدیک‌ترین همسایه بود.

ج)

[فایل پیاده‌سازی این قسمت: pr5_wdknn.ipynb]

الگوریتم kNN برای دسته‌بندی را به صورت زیر نیز می‌توان نشان داد:

$$y_t' = \arg\max_{c\in\{c_1,c_2\}} \sum_{x_i\in\phi(x_t)} I(y_i = c)$$
$$= \max\left\{ \sum_{x_i\in(x_t)} I(y_i = c_1), \sum_{x_i\in(x_t)} I(y_i = c_2) \right\}$$

که در آن I تابع ارضا شرط با خروجی ۱و۰ است.

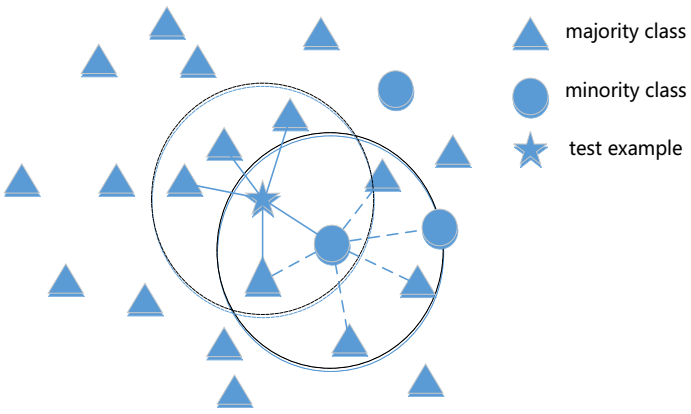
ایده الگوریتم WDkNN وزن‌دهی به مجموع‌های بالا است. به صورتی که نقاط نزدیک‌تر دارای وزن بیشتری باشند:

$$y_t' = \arg\max_{c\in\{c_1,c_2\}} \sum_{x_i\in\phi(x_t)} I(y_i = c)\frac{1}{d(x_t,x_i)}$$

که در آن تابع d فاصله اقلیدسی بین داده‌هاست. بنابرین در این الگوریتم همسایه‌های نزدیک‌تر بیشتر مورد توجه قرار خواهند گرفت.

د)

روش پیشنهادی مقاله بر پایه روش WDkNN برای مجموعه داده‌های نامتوازن توسعه داده شده است. به این صورت که ابتدا کلاس اقلیت و اکثریت را بر اساس تعداد داده‌های موجود برای هر کلاس محاسبه می‌کند. پس از پیدا کردن k نزدیکترین همسایه هدف این الگوریتم در نظر گرفتن ویژگی‌های محلی نقاط اقلیت و توزیع اطراف آن‌ها برای بهبود کارایی الگوریتم می‌باشد (شکل زیر). به این صورت که هر چه تعداد کلاس‌های اکثریت اطراف نمونه اقلیت بیشتر باشد وزن این نمونه بیشتر خواهد شد.



بنابرین دسته‌بند به صورت زیر خواهد شد:

$$y_t^{'} = \arg\max_{c\in\{c_1,c_2\}} \sum_{x_i\in\phi(x_t)} I(y_i = c)\frac{1}{d(x_t,x_i)} w(L)$$

که در آن $w(l)$ برای کلاس اکثریت یک و برای کلاس اقلیت به صورت زیر خواهد بود:

$$w(L) = (L_{\min} + 1)\lambda$$

که در آن لاندا برای نرمالسازی وزن‌هاست که در این‌جا ۱ در نظر گرفت شده است و همچنین L_{min} به صورت زیر محاسبه می‌شود:

$$L_{\min} = \frac{(N_{maj})^{\alpha}}{K}$$

که در آن N_{maj} تعداد نقاط اقلیت در k همسایگی نمونه اقلیت برای محاسبه وزن نهایی نمونه اقلیت است. پارامتر آلفا با توجه به نامتوازنی محیط مقداردهی می‌شود و برای محیط‌های خیلی نامتوازن‌تر مقداری بیشتر می‌گیرد تا تاثیر نمونه اقلیت بیشتر شود.

۵)

برای بررسی دو مدل پیشنهاد شده آزمایش‌هایی با ویژگی‌های زیر بر روی هفت مجموعه داده انجام شد. به ازای هر مدل مراحل زیر انجام شد:

- داده‌ها نرمال شدند.
- معیارهای Precision، Recall، F۱-measure و G-mean استخراج شدند.
- برای هر مجموعه داده برای k از ۱ تا ۵ معیارهای دقت استخراج شدند و از آن‌ها میانگین گرفته شد.
- برای هر مقدار k آزمایش به صورت ۵-fold انجام شد و از معیارها میانگین گرفته شد.
- برای بررسی اجمالی دو مدل از معیارهای به دست آمده ۷ مجموعه داده میانگین گرفته شد.

نتایج به دست آمده برای WDkNN:

Dataset	Precision	Recall	F1-measure	G-mean
ecoli0267vs35	0.904	0.63	0.69815873	0.75958558
ecoli3	0.66568254	0.55428571	0.57289909	0.719467
yeast0256vs3789	0.65661892	0.53052632	0.57678239	0.71250025
yeast02579vs368	0.80701996	0.78736842	0.79326046	0.87725984
yeast0359vs78	0.55655556	0.36	0.42224178	0.58190214
yeast2vs4	0.87031313	0.692	0.75373367	0.82123751
yeast3	0.76308734	0.7075	0.73030912	0.8274564
Overall	**0.74618249**	**0.60881149**	**0.64962646**	**0.75705839**

نتایج به دست آمده برای روش پیشنهادی:

Dataset	Precision	Recall	F1-measure	G-mean
ecoli0267vs35	0.878	0.67	0.72498413	0.7981435
ecoli3	0.63280952	0.57142857	0.5639019	0.72685806
yeast0256vs3789	0.63507433	0.54526316	0.57901207	0.72219889
yeast02579vs368	0.80033339	0.80210526	0.79793325	0.88495269
yeast0359vs78	0.51968254	0.384	0.42878342	0.59984799
yeast2vs4	0.85172294	0.712	0.76162378	0.83296637
yeast3	0.75232686	0.7175	0.73064641	0.83229176
Overall	0.72427851	0.62889957	0.65526928	0.77103704

روش پیشنهای برای اکثر مجموعه‌های داده مورد استفاده نتایج بهتری داشته است. همچنین با توجه به نتایج به دست آمده مشاهده می‌شود که روش پیشنهادی به صورت کلی بهتر عمل می‌کند. زیرا تمامی مجموعه داده‌های استفاده شده غیرمتوازن هستند و این الگوریتم برای داده‌های اقلیت وزن بیشتری را در نظر گرفته است.

همچنین این روش همانطور که انتظار می‌رود Precision پایین‌تر و Recall بالاتری نسبت به روش WDkNN دارد. زیرا در تمامی این مجموعه داده‌ها اقلیت کلاس مثبت بوده و بهبود روش این بوده است که این روش مقدار بیشتر از داده‌های مثبت را درست پیش‌بینی می‌کند، در نتیجه Recall بالاتر خواهیم داشت.

بخش دوم

-۱

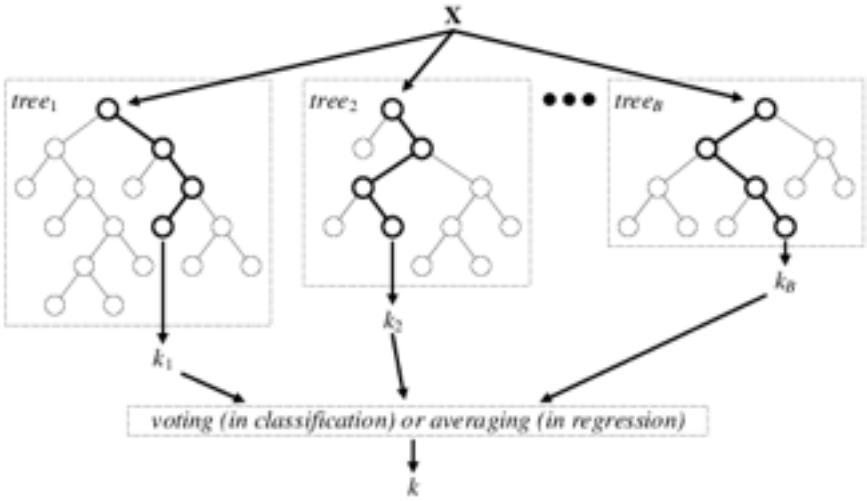
گر هرس کردن درخت تصمیم را در نظر نگیریم در نتیجه چون درخت تصمیم سعی دارد تمام نقاط را برازش کند در نتیجه حتی اگر مدل ساده هم باشد ممکن است عمق درخت زیاد شود و این خاصیت برازش تمام دادهها برای درخت تصمیم هم ساختار درخت را پیچیدهتر میکند هم در برخورد با دادههای تست ممکن است عملکرد خوبی نداشته باشد. ضمن اینکه دادههای آموزشی دارای نویز میباشند و چون درخت تصمیم نویز پذیر است لذا برای برازش دادههای نویزی، عمق درخت بالاجبار زیاد شده و درنتیجه ساختار پیچیده و خطا نیز زیاد میشود. گاهی اوقات مدلها نیز پیچیده هستند در نتیجه برای برازش کردن آنها نیازمند ساختار پیچیدهای نیز میباشد که این کار کارایی درخت تصمیم را برای دادههای تست پایین میآورد.

-۲

همانطور که در قسمت قبل توضیح داده شد وجود دادههای نویزی و ساختار پیچیده دادهها باعث پیچیدگی ساختار درخت و زیاد شدن عمق درخت میباشد برای رفع این مشکل میتوان درخت تصمیم را هرس کرد. هرس کردن میتواند شامل یک زیر درخت یا برگ باشد و نحوه عملکرد آن به این صورت است که اگر قصد داشته باشیم یک زیر درخت را با یک برگ جایگزین کنیم، در آن زیر درخت تعداد کلاسهای مثبت و منفی را مقایسه میکنیم و هرکدارم که بیشتر بود به عنوان جایگزین درخت انتخاب میکنیم.

-۳

جنگل تصادفی درخت تصمیم‌های زیادی را تولید می‌کند. برای طبقه‌بندی یک شیء جدید از برداری ورودی را در انتهای هر یک از درختان جنگل تصادفی قرار می‌دهد. هر درخت به ما یک طبقه‌بندی می‌هد و می‌گوییم این درخت به آن کلاس "رای" می‌دهد. جنگل طبقه‌بندی‌ای که بیشترین رای را داشته باشد (بین همه درخت های جنگل) انتخاب می‌کند.



هر درخت به صورت زیر تشکیل می‌شود:

۱. اگر N تعداد حالت‌ها در مجموعه داده‌های train (مجموعه‌ی کار) باشد، N حالت را به صورت تصادفی با جایگذاری از داده‌های اصلی، نمونه‌گیری می‌کنیم.این نمونه مجموعه‌ی کار برای این درخت می‌باشد.

۲. اگر M متغیر داشته باشیم و m را کوچکتر از M در نظر بگیریم به طوری که در هر گره، m متغیر به صورت تصادفی از M انتخاب می‌شوند و بهترین جداسازی روی این m متغیر برای جداسازی گره استفاده می‌شود. مقدار m در طول ساخت جنگل ثابت در نظر گرفته می‌شود.

۳. هر درخت به اندازه‌ی ممکن بزرگ می‌شود. هیچ هرسی وجود ندارد.

نرخ خطای جنگل به دو مورد زیر بستگی دارد:

- همبستگی بین هر دو درخت در جنگل. افزایش همبستگی نرخ خطای جنگل را افزایش می‌دهد.
- قدرت هر یک از درختان در جنگل. هر درخت با نرخ خطای کم یک طبقه بند قوی است. افزایش قدرت هر یک از درختان نرخ خطای جنگل را کاهش می‌دهد.

کاهش m هم همبستگی و هم قدرت را کاهش می‌دهد. و افزایشش هر دو را افزایش می‌دهد.

ویژگی‌های جنگل تصادفی

- در میان الگوریتم‌های فعلی از نظر دقت بی‌نظیر است.
- روی داده‌های بسیار بزرگ قابل اجراست.
- می‌تواند هزاران متغیر را بدون حذف متغیرها مدیریت کند.
- برآوردی از مهمترین متغیرها در طبقه‌بندی می‌دهد.
- راه کارایی برای برآورد داده‌ها گم‌شده دارد.

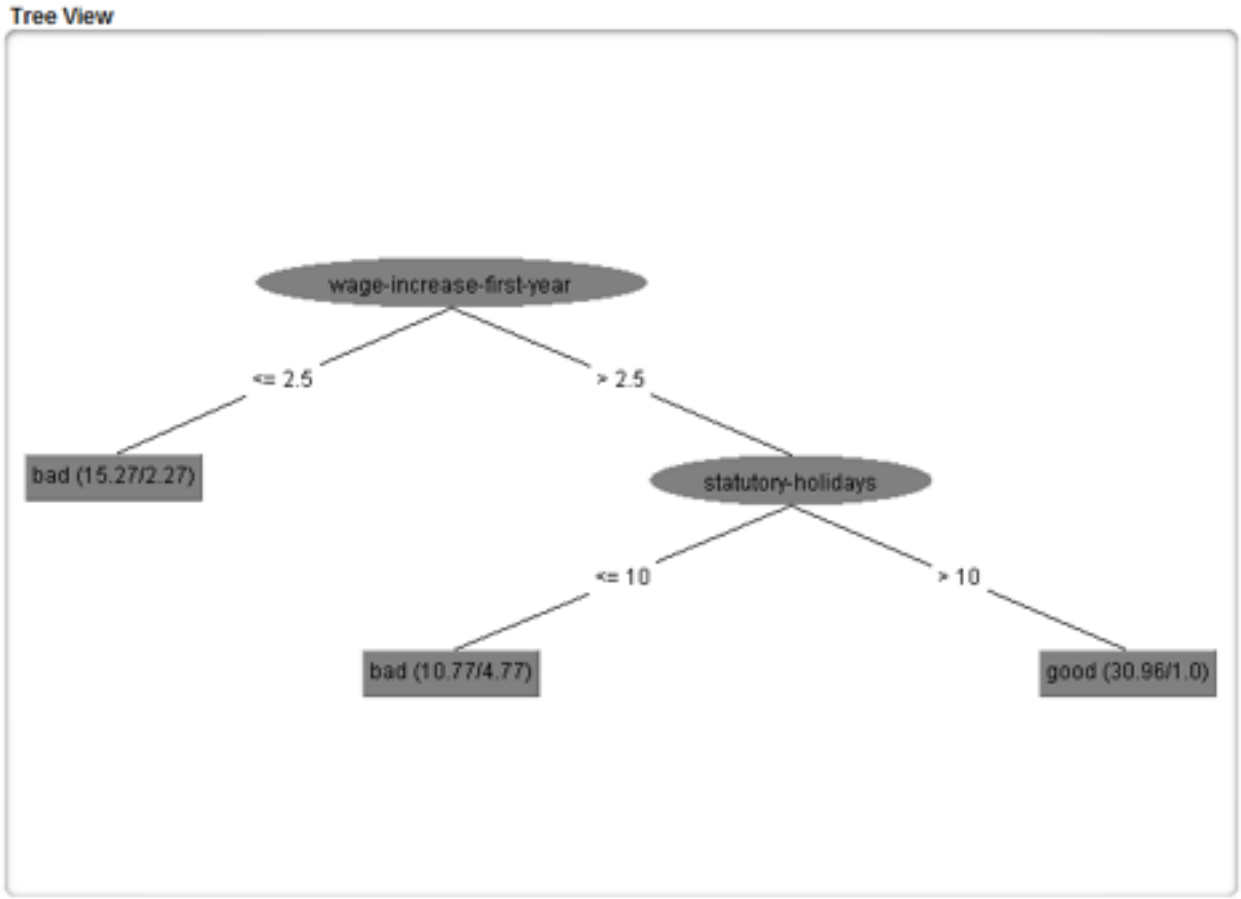
-۴

قسمت اول

Confusion Matrix

	HYPOTHESIS OUTPUT(GOOD)	HYPOTHESIS OUTPUT(BAD)
True Class good	**28**	**9**
True Class bad	**6**	**14**

True Positive = 28 , False Positive = 6 , True Negative = 14 , False Negative = 9



ابتدا از ریشه شروع میکنیم. با توجه به ویژگی wage-increase-first-year که برای داده مورد نظر برابر ۳ است در نتیجه وارد شاخه سمت راست میشود. ویژگی بعدی که به آن برخورد میکنیم statutory-holidays است که برای داده مورد نظر برابر ۱۲ است در شاخه سمت راست به یک برگ برخوردیم و در نتیجه کلاس داده مورد نظر good خواهد بود.

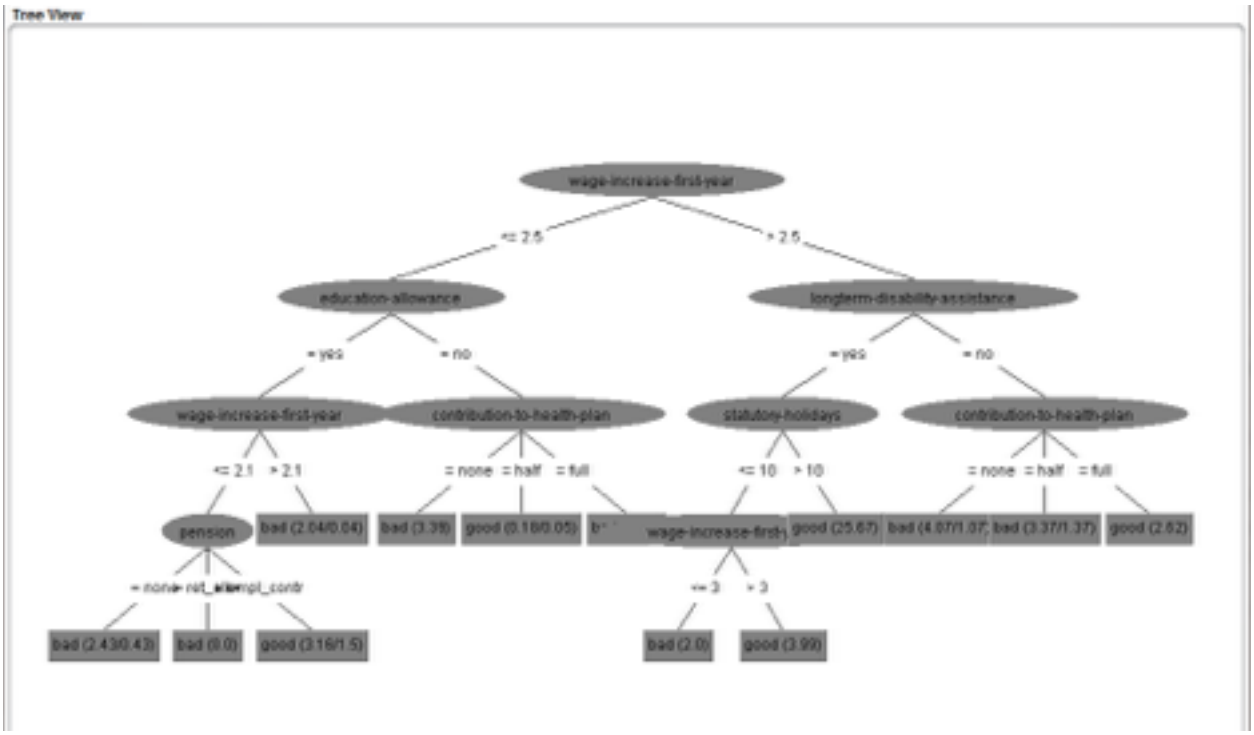
قسمت دوم

پارامتر unpruned برای هرس کردن درخت به همان شیوه ای که قبلا توضیح دادیم استفاده میشود.

Confusion Matrix

	HYPOTHESIS OUTPUT(GOOD)	HYPOTHESIS OUTPUT(BAD)
True Class good	**31**	**6**
True Class bad	**6**	**14**

True Positive = 31 , False Positive = 6 , True Negative = 14 , False Negative = 6



مانند قبل مقدار ویژگی که در ریشه قرار دارد را در داده تست پیدا کرده که wage-increase-first-year است و با ۲.۵ مقایسه می کنیم که این ویژگی برای داده تست برابر ۳ است در نتیجه حرکت به سمت شاخه سمت راست خواهد بود حال ویژگی longterm-disability-assistance را بررسی میکنیم که برای داده تست yes است بنابراین شاخه سمت چپ را بررسی میکنیم. ریشه زیردرخت جدید دارای ویژگی statutory-holiday است که برای داده تست برابر بنابراین وارد شاخه سمت راست میشویم که به برگ برخورد میکنیم که کلاس good را نشان میدهد بنابراین در کلاس good قرار میگیرد.

در مقایسه با درخت قبلی چون این درخت هرس نشده دارای عمق بیشتری میباشد و دادههای آموزش را بهتر برازش کرده است در نتیجه برای این داده تست نتیجه بهتری داده است.

قسمت سوم

Confusion Matrix

	PREDICT GOOD	**PREDICT BAD**
Actual good	**35**	**2**
Actual bad	**4**	**16**

True Positive = 35 , False Positive = 4 True Negative = 16 , False Negative = 2

قسمت چهارم

بنابرین با توجه به نتایج مشخص است که جنگل تصادفی بهتر عمل کرده است و دقت نهایی بهتر بوده است. دلیل این امر این است که جنگل تصادفی چند بار درخت تصمیم را از ریشه به برگ میسازد و سپس برای دستهبندی داده تست رایگیری انجام میدهد که این امر در افزایش دقت جنگل تصادفی موثر است زیرا احتمال خطا کاهش مییابد. این روش شبیه روش ensemble عمل میکند با این تفاوت که مدلها همه از نوع درخت تصمیم هستند.