۱.

وقتی داده ها خطی جدایی پذیر نباشند، باید از کلاس بندیهایی با مرزهای تصمیم گیری غیرخطی برای کلاس بندی داده ها استفاده کنیم.یکی از این راه‌ها که برای دسته‌بندی چنین داده هایی استفاده می شود، استفاده از توابع هسته (kernel function) برای گسترش فضاست. یکی از مزیت های استفاده از تابع‌های کرنل برای گسترش فضا در مقایسه با استفاده از توان های بالاتر ویژگی ها (features)، مزیت محاسباتی است که در حالت استفاده از کرنل تنها نیاز به محاسبه $\binom{n}{2}$ تابع کرنل داریم. اما وقتی برای گسترش فضا از توان های چندم ویژگی ها استفاده می کنیم، ممکن است محاسبات بسیار زیاد شود.

۱.توابع هستهٔ خطی(Linear Kernel):

تابع هستهٔ خطی از اساسی‌ترین نوع توابع هسته‌ای می‌باشند.ماهیت آن معمولا یک بعدی است.در مواقعی که تعداد فیچرها زیاد است، اثبات می‌شود که تابع هستهٔ خطی بهترین عملکرد را دارد.در مسائل کلاس‌بندی متن(text-classification problems) کاربرد دارد زیرا بسیاری از کلاس‌بندی های از این قبیل به صورت خطی جدایی‌پذیر می‌باشند.توابع هستهٔ خطی از سایر توابع سریع‌تر می‌باشند.

فرمول:

F(x, xj) = sum(x.xj)

۲.توابع هستهٔ چند جمله‌ای(Polynomial Kernel):

استفاده از توابع هسته چند جمله ای با درجه d>1 در کلاس بندهای بردار پشتیبان می تواند به انعطاف مرزهای تصمیم گیری کمک کند. گویا ما داده ها را به فضای d بعدی برده ایم و در آنجا کلاس بندی خطی تعریف کرده ایم که در فضای اصلی مرز تصمیم گیری غیرخطی دارد.

$$K(x_i, x_i') = \left(1 + \Sigma_{j=1}^{p} x_{ij} x_{ij}'\right)^d$$

۳.توابع هستهٔ شعاعی(Radial Kernel)

این تابع هسته معمولا عملکرد خوبی در جداسازی داده ها به دو کلاس دارد. علت این عملکرد خوب آن است که مثلا فرض کنید می خواهیم یک داده تست $x^* = \left(x_1^*, .., x_p^*\right)$ را کلاس بندی کنیم. اگر داده تست از داده x_i دور باشد، فاصله اقلیدسی $\Sigma_{j=1}^{p}\left(x_{ij} - x_j^*\right)^2$ زیاد و بنابراین $K(x_i, x^*)$ کم خواهد شد. پس نقش x_i در تابع $f(x^*) = \beta_0 + \Sigma_{i \in S} \alpha_i K(x_i, x^*)$ کم خواهد بود. بنابراین تابع هسته شعاعی به صورت محلی عمل می کند. به این معنا که هر داده ای که به داده تستمان نزدیک تر باشد، تاثیر بیشتری در تصمیم گیری ما دارد.

$$K(x_i, x_i') = \exp\left(-\gamma \Sigma_{j=1}^{p}\left(x_{ij} - x_{ij}'\right)^2\right)$$

.۴

Gaussian Radial Basis Function (RBF)

یکی از بهترین و شناخته‌شده‌ترین توابع هسته در SVM است.معمولا برای داده‌های غیرخطی انتخاب می‌شود.در صورت نداشتن شناخت قبلی از داده‌ها، به جداسازی مناسب کمک می‌کند.

$$F(x, xj) = \exp(-gamma * ||x - xj||^2)$$

مقدار گاما از ۰ تا ۱ متغیر است و بصورت دستی تعیین می‌شود.

.۲

Name of the classifier: Classifier

Kernel =rbf

Gamma=1

Random_state=1

```
Confusion Matrix:  [[  0 205   0   0]
 [  0 187   0   0]
 [  0 199   0   0]
 [  0 209   0   0]]
Accuracy :  23.375
Report :               precision    recall  f1-score   support

     class 0       0.00      0.00      0.00       205
     class 1       0.23      1.00      0.38       187
     class 2       0.00      0.00      0.00       199
     class 3       0.00      0.00      0.00       209

    accuracy                           0.23       800
   macro avg       0.06      0.25      0.09       800
weighted avg       0.05      0.23      0.09       800
```

۳.

Name of the classifier: Classifier1

Kernel function=rbf

```
Confusion Matrix:  [[189  16   0   0]
 [ 15 153  19   0]
 [  0  19 175   5]
 [  0   0  21 188]]
Accuracy :  88.125
Report :                precision    recall  f1-score   support

     class 0       0.93      0.92      0.92       205
     class 1       0.81      0.82      0.82       187
     class 2       0.81      0.88      0.85       199
     class 3       0.97      0.90      0.94       209

    accuracy                           0.88       800
   macro avg       0.88      0.88      0.88       800
weighted avg       0.88      0.88      0.88       800
```

Name of the classifier: Classifier2

Kernel function: linear

Random_state=1

Gamma=scale

```
Confusion Matrix:  [[201   4   0   0]
 [ 10 173   4   0]
 [  0  11 182   6]
 [  0   0   3 206]]
Accuracy :  95.25
Report :                precision    recall  f1-score   support

     class 0       0.95      0.98      0.97       205
     class 1       0.92      0.93      0.92       187
     class 2       0.96      0.91      0.94       199
     class 3       0.97      0.99      0.98       209

    accuracy                           0.95       800
   macro avg       0.95      0.95      0.95       800
weighted avg       0.95      0.95      0.95       800
```

Name of the classifier: Classifier3

Kernel function=sigmoid

Random_state=1

```
Confusion Matrix:  [[189  16   0   0]
 [  9 169   9   0]
 [  0  17 176   6]
 [  0   0  14 195]]
Accuracy :  91.125
Report :                precision    recall  f1-score   support

     class 0       0.95      0.92      0.94       205
     class 1       0.84      0.90      0.87       187
     class 2       0.88      0.88      0.88       199
     class 3       0.97      0.93      0.95       209

    accuracy                           0.91       800
   macro avg       0.91      0.91      0.91       800
weighted avg       0.91      0.91      0.91       800
```

Name of the classifier: Classifier4

Kernel function=linear

C=0.05

Random_state=1

```
Confusion Matrix:  [[198   7   0   0]
 [  9 168  10   0]
 [  0  12 183   4]
 [  0   0   9 200]]
Accuracy :  93.625
Report :                precision    recall  f1-score   support

     class 0       0.96      0.97      0.96       205
     class 1       0.90      0.90      0.90       187
     class 2       0.91      0.92      0.91       199
     class 3       0.98      0.96      0.97       209

    accuracy                           0.94       800
   macro avg       0.94      0.94      0.94       800
weighted avg       0.94      0.94      0.94       800
```

Name of the classifier: Classifier5

Kernel function= poly

```
Confusion Matrix:  [[157  48   0   0]
 [ 15 145  27   0]
 [  0  47 144   8]
 [  0   1  45 163]]
Accuracy :  76.125
Report :               precision    recall  f1-score   support

     class 0       0.91      0.77      0.83       205
     class 1       0.60      0.78      0.68       187
     class 2       0.67      0.72      0.69       199
     class 3       0.95      0.78      0.86       209

    accuracy                           0.76       800
   macro avg       0.78      0.76      0.77       800
weighted avg       0.79      0.76      0.77       800
```

با توجه به نتایج بدست آمده کلاسبند با تابع هستهٔ خطی(classifier2)، بهترین عملکرد و کلاسبند با تابع هستهٔ چندجمله‌ای(classifier5) بدترین عملکرد را روی داده‌ها دارد و این نشان می‌دهد داده‌ها بطور خوبی خطی جدایی پذیر هستند.

همچنین در دو کلاسبند classifier2 و classifier4 که هر دو از تابع هستهٔ خطی استفاده می‌کنند classifier2عملکرد بهتری دارد. در classifier4، c=0.05 و در classifier2 مقدار c بصورت پیش‌فرض ۱ است. هر چه پارامترc بزرگتر باشد حاشیهٔ صفحه‌ای که داده‌ها را از هم جدا می‌کند کمتر است و در نتیجه از کلاس‌بندی نادرست داده‌ها جلوگیری می‌کند. البته مقادیر بزرگ این پارامتر موجب overfit شدن مدل نیز می‌شوند.

۴.

Name of the classifier :Svm_soft_margin

Kernel=linear

C=0.01

```
Confusion Matrix:  [[192  13   0   0]
 [  6 166  15   0]
 [  0  17 179   3]
 [  0   0  25 184]]
Accuracy :  90.125
Report :                precision    recall  f1-score   support

     class 0       0.97      0.94      0.95       205
     class 1       0.85      0.89      0.87       187
     class 2       0.82      0.90      0.86       199
     class 3       0.98      0.88      0.93       209

    accuracy                           0.90       800
   macro avg       0.90      0.90      0.90       800
weighted avg       0.91      0.90      0.90       800
```

Name of the classifier: Svm_margin(hard margin)

Kernel=linear

C=1

```
Confusion Matrix:  [[201   4   0   0]
 [ 10 173   4   0]
 [  0  11 182   6]
 [  0   0   3 206]]
Accuracy :  95.25
Report :                precision    recall  f1-score   support

     class 0       0.95      0.98      0.97       205
     class 1       0.92      0.93      0.92       187
     class 2       0.96      0.91      0.94       199
     class 3       0.97      0.99      0.98       209

    accuracy                           0.95       800
   macro avg       0.95      0.95      0.95       800
weighted avg       0.95      0.95      0.95       800
```

در مورد پارامترc در سوال قبل توضیح دادهشد.در این مثال hard marginعملکرد بهتری از soft margin داشته است.البته معیار ارزیابی validation setبوده و دادهی تست ما محدود و تصادفی است بهتتر بود از k-fold cv استفاده می شد تا اگر به ازای مقادیر بالای c ، overfitting داشتیم آن را تشخیص دهیم.

۵.

ب)

بسیاری از الگوریتم‌های یادگیری ماشین نمی‌توانند با داده‌های کیفی(categorical) به طور مستقیم کار کنند.به همین دلیل از one hot encoding استفاده می‌کنیم.هم‌چنین این کدگذاری این اطمینان را به ما می‌دهد که یادگیری ماشین متصور نخواهد شد که اعداد بالاتر از اهمیت بیشتری برخوردار هستند.به عنوان مثال مقدار ۸ از مقدار ۱ بزرگتر است اما این به این معنا نیست که اهمیت آن نیز بیشتر است.

ج)

تبدیل داده فرآیند تغییر فرمت، ساختار و یا مقادیر داده است.تبدیل داده مزایای زیادی دارد:

با تبدیل داده سازماندهی آن بهتر می‌شود.استفاده از داده‌ی تبدیل شده برای انسان و کامپیوتر آسان تر است.

استفاده از تبدیلات مناسب کیفیت داده را بهبود می‌بخشد و برنامه‌هایی که از داده استفاده می کنند را از مشکلات احتمالی (داده‌های پوچ، اندیس گذاری غلط، تکرار های غیر منتظره، فرمت های ناسازگار) محافظت می‌کنند.درکل تبدیل داده سازگاری بین برنامه‌ها، سیستم‌ها و انواع داده را تسهیل می‌بخشد.ممکن است لازم باشد داده‌های مورد استفاده برای اهداف مختلف به روشهای مختلفی تغییر شکل داده شوند.

Log transformation

از این تبدیل وقتی می توان استفاده کرد که توزیع داده‌ای بسیار کج باشد.

۶.

در قسمت ۵.الف بر روی فیچر battery_power روش binning بصورت زیر اعمال شده است:

مقدار مینیمم این فیچر 1.678397685369917-و مقدار ماکسیمم آن 1.728379683602883 است.

به کمک تابع np.linspace() نقاط مرزی bin ها را بدست می‌آوریم :

```
array([-1.67839769, -0.54280523,  0.59278723,  1.72837968])
```

که دو سر بازه همان مینیمم و ماکسیمم این ویژگی اند.

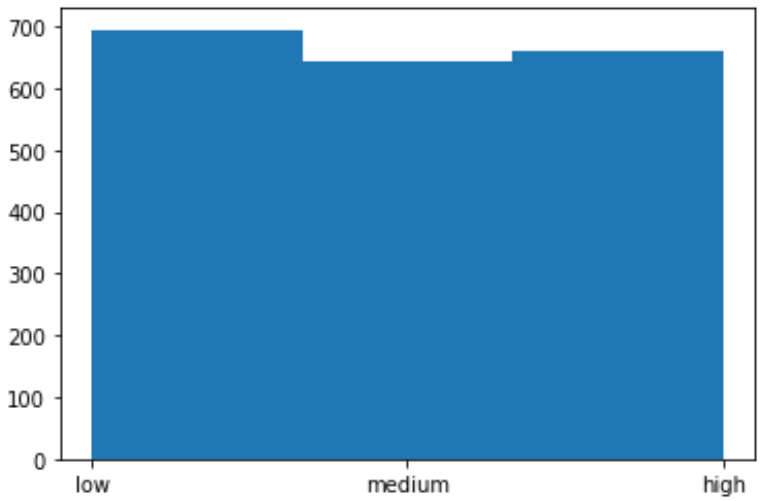
حال با توجه به تابع فوق به هر یک از نمونه‌های این ستون یک label نسبت می دهیم:

اگر مقدار نمونه بین 0.54280523-و 1.67839769- باشد:low

اگر مقدار نمونه بین 0.59278723و0.54280523- باشد:medium

اگر مقدار نمونه بین 1.72837968 و 0.59278723 باشد:high

نمودار زیر توزیع هر یک از این labelها را نشان می‌دهد:



در قسمت ۵.ب روی تنها فیچر کتگوریکال (battery_power)، one hot encoding را اعمال کردیم.دیتاست جدید را با کلاسبند classifier2 ، کلاسبندی می‌کنیم همانطور که مشاهده می‌کنید نتیجه اندکی بدتر شده‌است:

```
Confusion Matrix:  [[199   6   0   0]
 [ 13 167   7   0]
 [  0  10 181   8]
 [  0   0  13 196]]
Accuracy :  92.875
Report :               precision    recall  f1-score   support

     class 0       0.94      0.97      0.95       205
     class 1       0.91      0.89      0.90       187
     class 2       0.90      0.91      0.91       199
     class 3       0.96      0.94      0.95       209

    accuracy                           0.93       800
   macro avg       0.93      0.93      0.93       800
weighted avg       0.93      0.93      0.93       800
```

درقسمت ۵.ج از Quantile Transformer Scaler استفاده شده‌است.

دیتاست جدید را با کلاسبند classifier ، کلاسبندی می‌کنیم همانطور که مشاهده می‌کنید نتیجه به طور قابل ملاحظه‌ای بهتر شده‌است:(از accuracy=0.23 به accuracy=0.76)

=

```
Confusion Matrix:  [[174  31   0   0]
 [ 26 137  24   0]
 [  1  41 146  11]
 [  0   1  50 158]]
Accuracy :  76.875
Report :               precision    recall  f1-score   support

     class 0       0.87      0.85      0.86       205
     class 1       0.65      0.73      0.69       187
     class 2       0.66      0.73      0.70       199
     class 3       0.93      0.76      0.84       209

    accuracy                           0.77       800
   macro avg       0.78      0.77      0.77       800
weighted avg       0.78      0.77      0.77       800
```

در قسمت ۵.د ستون جدید superficial را که مساحت سطح گوشی است را ساختیم و دو فیچر px_heightو px_width را حذف کردیم.

دیتاست جدید را با کلاسبند classifier2 ، کلاسبندی می‌کنیم همانطور که مشاهده می‌کنید نتیجه بدتر شده‌است:

```
Confusion Matrix:  [[187  18   0   0]
 [ 19 145  23   0]
 [  0  28 145  26]
 [  0   0  27 182]]
Accuracy :  82.375
Report :               precision    recall  f1-score   support

     class 0       0.91      0.91      0.91       205
     class 1       0.76      0.78      0.77       187
     class 2       0.74      0.73      0.74       199
     class 3       0.88      0.87      0.87       209

    accuracy                           0.82       800
   macro avg       0.82      0.82      0.82       800
weighted avg       0.82      0.82      0.82       800
```

یکبار هم همهٔ تغییرات سوال ۵ را بر دیتاست اعمال می کنیم:

دیتاست جدید را با کلاسبند classifier ، کلاسبندی می‌کنیم همانطور که مشاهده می‌کنید نتیجه به طور خوبی بهتر شده‌است:(از accuracy=0.23 به accuracy=0.69)

```
Confusion Matrix:  [[153  52   0   0]
 [ 30 122  34   1]
 [  0  55 126  18]
 [  0   1  56 152]]
Accuracy :  69.125
Report :               precision    recall  f1-score   support

     class 0       0.84      0.75      0.79       205
     class 1       0.53      0.65      0.59       187
     class 2       0.58      0.63      0.61       199
     class 3       0.89      0.73      0.80       209

    accuracy                           0.69       800
   macro avg       0.71      0.69      0.70       800
weighted avg       0.72      0.69      0.70       800
```

۷.

بطور کلی تفاوت الگوریتم‌های ساخت درخت در انتخاب معیارتقسیم است و تعداد تقسیم بندی هایی است

که در هر مرحله انجام می‌دهند:

برخی از معیارهای تقسیم:

۱.نسبت خطای کلاس‌بندی(Classification error rate)

تابع خطای زیر برای بررسی نسبت تخصیص کلاس‌ها می‌باشد:

$$E = 1 - \max_k \hat{p}_{mk}$$

که $\hat{p}_{mk}$ نسبت تخصیص داده های آموزشی واقع در m امین ناحیه به کلاس k ام است.

۲. اندیس جینی

(Gini index) که طبق فرمول زیر محاسبه می شود:

$$G = \Sigma_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

این مقدار، میزان واریانس بین k کلاس را بررسی می کند. اگر $\hat{p}_{mk}$ نزدیک به صفر یا نزدیک به ۱ باشد، اندیس جینی نزدیک به صفر خواهد بود (مقدار بهینه می شود) . هر چه اندیس جینی کمتر باشد، یعنی در آن ناحیه/برگ، اغلب داده ها به یک کلاس تعلق دارند.این معیار، به نوعی میزان خلوص(Purity) هر برگ را می سنجد.

۳. خطای انتروپی متقاطع (cross entropy)

تقریبا معادل اندیس جینی است:

$$D = -\Sigma_{k=1}^{m} \hat{p}_{mk} \log \hat{p}_{mk}$$

از آنجاییکه $0 < \hat{p}_{mk} < 1$ ، پس $-\hat{p}_{mk} \log \hat{p}_{mk} > 0$ است. بنابراین وقتی $\hat{p}_{mk}$ نزدیک به صفر یا ۱ باشد، خطای انترپی نزدیک به صفر خواهد شد. بنابراین مشابه اندیس جینی ، وقتی این خطا مینیمم می شود که اعضای یک دسته همگی برچسب مشابهی داشته باشند.

الگوریتم ID3(Iterative Dichotomiser 3)

الگوریتم در هر مرحله ویژگی‌ها را به دو یا چند بخش جدید تقسیم می‌کند.این الگوریتم از یک روش حریصانه از بالا به پایین برای ساختن درخت تصمیم استفاده می‌کند.یعنی از بالا شروع به ساختن درخت می‌کند و در هر تکرار بهترین ویژگی را برای ایجاد گره انتخاب می‌کند.

مراحل:

۱.محاسبهٔInformation Gain برای هر ویژگی.

۲. مجموعه داده‌ها را با استفاده از ویژگی که حداکثر میزان Information Gain را دارد به زیرمجموعه‌هایی تقسیم می‌کنیم.

۳. با استفاده از ویژگی که حداکثر Information Gain را داشت یک گره درخت تصمیم را تشکیل می‌دهیم.

۴.اگر همه داده ها از یک کلاس بودند گره فعلی را بعنوان گره برگ و کلاس را بعنوان برچسب آن قرار می دهیم.

۵.مراحل بالا را تکرار می کنیم تا جایی که یا فیچرها تمام شود ویا درخت تصمیم دارای همهٔ گره‌های برگ باشد.

الگوریتم Card:

این الگوریتم مشابه الگوریتم Recursive binary splitting است با این تفاوت که معیار انتخاب ویژگی در این الگوریتم Gini Index است.

Train using Gini Index:

Name: Clf_gini

```
max_depth=3, min_samples_leaf=5
```

```
Confusion Matrix:  [[166  39   0   0]
 [ 16 144  27   0]
 [  0  45 106  48]
 [  0   0  28 181]]
Accuracy :  74.625
Report :               precision    recall  f1-score   support

     class 0       0.91      0.81      0.86       205
     class 1       0.63      0.77      0.69       187
     class 2       0.66      0.53      0.59       199
     class 3       0.79      0.87      0.83       209

    accuracy                           0.75       800
   macro avg       0.75      0.74      0.74       800
weighted avg       0.75      0.75      0.74       800
```

Name: Clf_gini2

```
max_depth=7, min_samples_leaf=5
```

```
Confusion Matrix:  [[183  22   0   0]
 [ 19 149  19   0]
 [  0  25 151  23]
 [  0   0  21 188]]
Accuracy :  83.875
Report :               precision    recall  f1-score   support

     class 0       0.91      0.89      0.90       205
     class 1       0.76      0.80      0.78       187
     class 2       0.79      0.76      0.77       199
     class 3       0.89      0.90      0.90       209

    accuracy                           0.84       800
   macro avg       0.84      0.84      0.84       800
weighted avg       0.84      0.84      0.84       800
```

Name: Clf_gini3

```
max_depth=3, min_samples_leaf=100
```

```
Confusion Matrix:  [[167  38   0   0]
 [ 26 134  27   0]
 [  0  45 122  32]
 [  0   0  46 163]]
Accuracy :  73.25
Report :               precision    recall  f1-score   support

     class 0       0.87      0.81      0.84       205
     class 1       0.62      0.72      0.66       187
     class 2       0.63      0.61      0.62       199
     class 3       0.84      0.78      0.81       209

    accuracy                           0.73       800
   macro avg       0.74      0.73      0.73       800
weighted avg       0.74      0.73      0.73       800
```

با توجه به نتایج افزایش عمق درخت نسبت به افزایش تعداد سمپل های برگ های درخت در بهبود نتایج تاثیری بیشتری دارد.

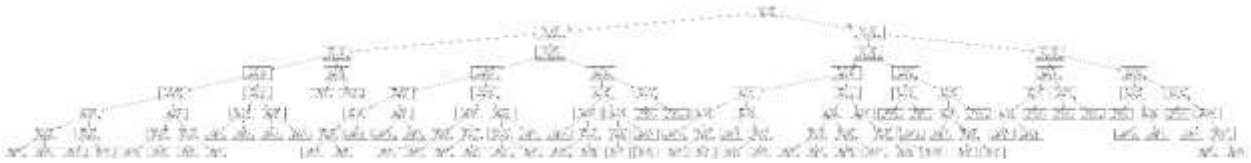
Train using Entropy:

Name: Clf_entropy

```
max_depth=3, min_samples_leaf=5
Confusion Matrix:  [[167  38   0   0]
 [ 26 134  27   0]
 [  0  45 122  32]
 [  0   0  46 163]]
Accuracy :  73.25
Report :               precision    recall  f1-score   support

     class 0       0.87      0.81      0.84       205
     class 1       0.62      0.72      0.66       187
     class 2       0.63      0.61      0.62       199
     class 3       0.84      0.78      0.81       209

    accuracy                           0.73       800
   macro avg       0.74      0.73      0.73       800
weighted avg       0.74      0.73      0.73       800
```

Visualizing the best decision tree

Name: Clf_gini2



۱۰.

الگوریتم های تشکیل درخت تصمیم اغلب منجر به تولید درختی می شوند که بر روی نمونه های آموزشی، جواب خوبی برمی گرداند اما بر داده‌های نمونه‌های تست، کارایی پایینی دارند. این امر به دلیل پیچیده شدن درخت (عمق زیاد و استفاده از انشعاب های متعدد) است. یک درخت کوچکتر با تعداد انشعاب های کمتر ممکن است مقدار کمی بایاس داشته باشد، اما به شدت واریانس کمتری بر روی داده های تست خواهد داشت. بنابراین یکی از راهکارها، هرس کردن(Pruning) درخت است.

یک ایده برای هرس کردن درخت این است که از بالا به پایین حرکت کنیم و در هر انشعابی که RSS نسبت به انشعاب قبلی مقدار کمی (کمتر از یک از حد آستانه ای) کاهش یافته بود، زیرشاخه های آن را از درخت حذف کنیم. این استراتژی چندان مناسب نیست زیرا کوتاه نگر(short sighting) است. چون ممکن است یک انشعاب چندان خوب نباشد اما در ادامه زیر شاخه های آن شامل انشعاب های خیلی خوبی باشند.

یک استراتژی بهتر می تواند تولید یک درخت T_0 بزرگ و بعد هرس کردن آن باشد. به طور شهودی ما به دنبال زیردرختی هستیم که دارای کمترین نرخ خطا بر روی داده های تست باشد، که نرخ خطای تست را می توانیم با cross- validation error تخمین بزنیم. چک کردن تمام زیر درخت های یک درخت و محاسبه cross- validation error برای همه آنها زمان‌بر است. در عوض، ما از روش زیر برای هرس کردن درخت استفاده می کنیم:
دنباله ای از درخت ها که با پارمتر نامنفی α اندیس‌گذاری شده اند را در نظر می‌گیریم. به ازای هر مقدار α، زیر درخت $T \subset T_0$ ای وجود دارد که مقدار تابع هزینه هرس زیر به ازای آن کمینه می شود:

$$\Sigma_{m=1}^{|T|}\Sigma_{i:x_i\in R_m}\left(y_i - \widehat{y_{R_m}}\right)^2 + \alpha|T|$$

که $|T|$ تعداد برگ های زیردرخت Tاست. پارامتر α، تعدیل‌کننده ای بین میزان پیچیدگی درخت و

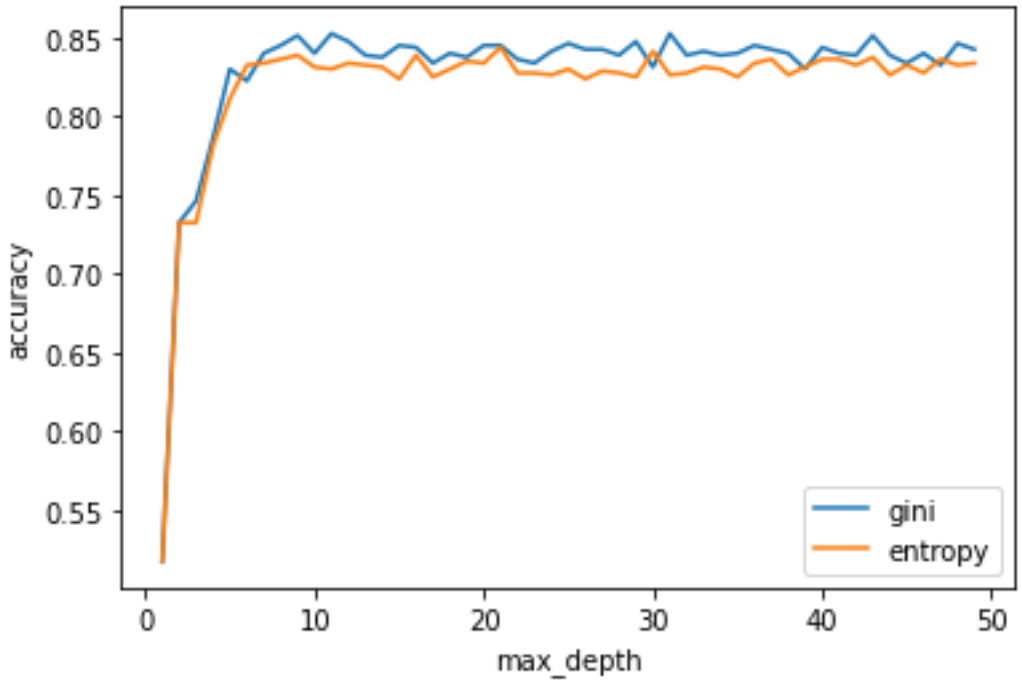
میزان وابستگی آن به داده های آموزشی است. وقتی $\alpha = 0$ زیر درخت T برابر همان درخت اولیه

است و هر چه مقدار α افزایش یابد، هرس بیشتری اعمال می شود. هر چه درخت بزرگتر می شود میزان جریمه تابع فوق به ازای برگ ها، بیشتر می شود و بنابراین تابع فوق به ازای زیردرخت های کوچکتری کمینه می شود.

.۱۱

Pre_prunning:

در این روش به ازای عمق های ۱ تا ۵۰ هم درخت های تصمیم با معیارGini Index ونیز Entropy را تشکیل دادیم و نتایج بصورت زیر بود:

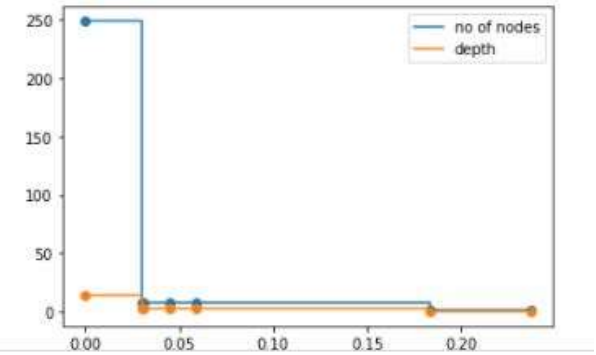


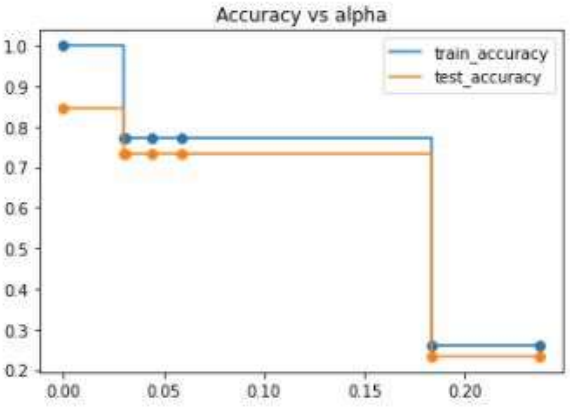
معیار Gini Index عمکرد بهتری داشته است هم چنین با مشاهدهٔ نمودار بهترین عمق درخت تقریبا ۹است.بنابر این بهترین درخت بصورت زیر خواهد بود:

Name: Clf_gini

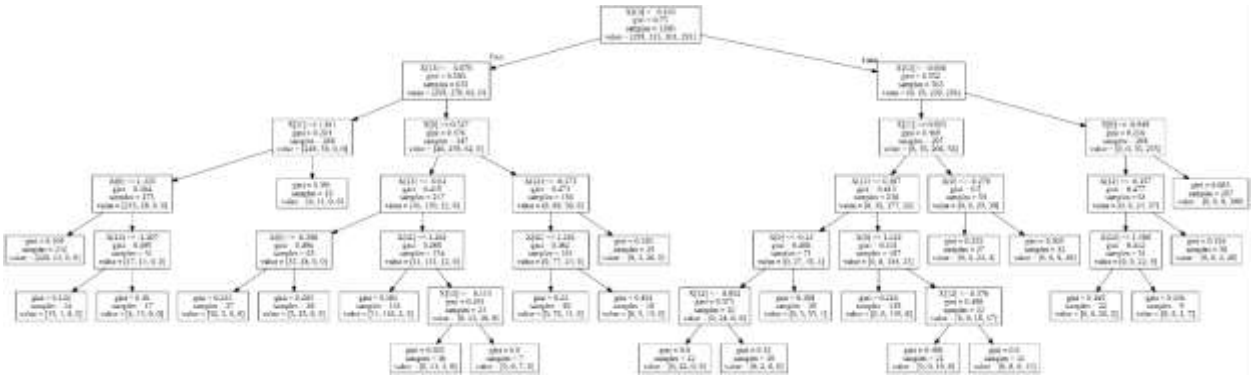
```
criterion="gini", max_depth=9
 Confusion Matrix:  [[187  18   0   0]
  [ 14 153  19   1]
  [  0  18 160  21]
  [  0   0  24 185]]
 Accuracy :  85.625
 Report :                precision    recall  f1-score   support

      class 0       0.93      0.91      0.92       205
      class 1       0.81      0.82      0.81       187
      class 2       0.79      0.80      0.80       199
      class 3       0.89      0.89      0.89       209

     accuracy                           0.86       800
    macro avg       0.86      0.85      0.86       800
 weighted avg       0.86      0.86      0.86       800
```

Post-prunning:






```
Train score 0.9008333333333334
Test score 0.82375
```



.۱۲

Name: Rfc

```
Confusion Matrix:  [[194  11   0   0]
 [ 15 147  25   0]
 [  0  32 151  16]
 [  0   0  15 194]]
Accuracy :  85.75
Report :                precision    recall  f1-score   support

     class 0       0.93      0.95      0.94       205
     class 1       0.77      0.79      0.78       187
     class 2       0.79      0.76      0.77       199
     class 3       0.92      0.93      0.93       209

    accuracy                           0.86       800
   macro avg       0.85      0.85      0.85       800
weighted avg       0.86      0.86      0.86       800
```

نتایج درخت تصمیم بعد از هرس کرن تقریبا با نتایج درخت تصمیم تصادفی یکسان است.

۱۳.

درخت های تصمیم برای مسائل رگرسیون و کلاس بندی دارای ویژگی های زیر هستند:

- ✓ درخت ها قابل توضیح به اکثر مردم هستند و بیان آنها راحت تر از مدل رگرسیون خطی است.
- ✓ درخت ها شمایی از عمل تصمیم گیری در ذهن انسان ها را نشان میدهد.
- ✓ درخت ها به صورت گرافیکی قابل تصویرسازی هستند و توسط افراد معمولی قابل تفسیر هستند.
- ✓ درخت ها به راحتی با متغیرهای کیفی کار می کنند بدون آنکه نیازی به معرفی متغیرهای شناسنده داشته باشند.
- ✗ متاسفانه دقت درخت ها در رگرسیون و کلاس بندی به اندازه سایر روش‌های معرفی شده نیست.

۱۴.

Rule induction حوزه‌ای از یادگیری ماشین است که در آن قوانین رسمی از مجموعه مشاهدات استخراج می‌شود.

:Ripper(Repeated Incremental Pruning to Produce Error Reduction)

الگوریتم Ripper یک الگوریتم کلاسبندی مبتنی قاعده است که مجموعه قوانینی را از مجموعه آموزش استخراج می‌کند.

موارد استفاده از الگوریتم Ripper:

۱.در مسائل کلاس‌بندی با توزیع نامتعادل داده روی کلاس‌ها به خوبی کار می‌کند.

۲. بر روی داده‌های noisy به خوبی عمل می‌کند زیرا از validation set برای جلوگیری از overfitting استفاده می‌کند.

عمکرد این الگوریتم به شیوه‌ی زیر است:

۱.حالت دو کلاسه:

در میان داده‌ها کلاس اکثریت را مشخص می‌کند(کلاسی که بیشتر داده‌ها برچسب آن را دارند)و این کلاس را بعنوان کلاس پیش‌فرض در نظر می‌گیرد.برای کلاس دیگر سعی در یادگیری و استخراج قوانین مخلف تشخیص آن کلاس دارد.

۲.حالت چند کلاسه:

تمام کلاس‌های موجود را بر اساس فروانی آن‌ها به ترتیبی خاص مرتب کنید(مثلا صعودی).

فرض کنید کلاس‌ها بصورت زیر مرتب شده‌اند:

C1, C2, C3, …., Cn

C1 – least frequent

Cn – most frequent

کلاس با حداکثر فراونی(Cn) به عنوان کلاس پیش‌فرض در نظر گرفته می‌شود.

نحوه‌ی استخراج قواعد بصورت زیر است:

در وهله اول قوانین برای آن دسته از سوابق که متعلق به کلاس C1 هستند استخراج می‌شود. داده ها با برچسب کلاس C1 به عنوان مثالهای مثبت+ve و سایر طبقات به‌عنوان مثالهای منفی -veدر نظر گرفته می‌شوند.

الگوریتم Sequential Covering برای تولید قوانینی استفاده می‌شود که بین مثال‌های +ve و -ve تمایز قائل می‌شوند. سپس سعی می کند قوانینی برای C2استخراج کند که آن را از کلاس‌های دیگر

متمایز کند. این فرآیند تا رسیدن به معیارهای توقف تکرار می‌شود، یعنی وقتی که ما با Cn(کلاس پیش فرض) باقی می مانیم.