به نام خدا

ریحانه قاسمیان روشن

1- هفت کرنل که در SVM زیاد استفاده میشوند :
1- کرنل خطی(Linear Kernel)
2- کرنل چندجمله ای (Polynomial Kernel)
3- کرنل RBF گوسی (Gaussian Radial Basis Function)
4- کرنل سیگموئید (Sigmoid Kernel)
5- کرنل گوسی (Gaussian Kernel)
6- کرنل تابع بسل (Bessel function kernel)
7- کرنل آنالیز واریانس (ANOVA kernel)

الگوریتم های SVM از گروهی از توابع ریاضی استفاده می کنند که به عنوان هسته شناخته می شوند. عملکرد یک هسته این است که داده را به عنوان ورودی نیاز دارد و آن را به شکل دلخواه تبدیل می کند. الگوریتم های مختلف SVM از انواع مختلف توابع هسته استفاده می کنند. این توابع انواع مختلفی دارند - به عنوان مثال ، خطی ، غیرخطی ، چند جمله ای ، تابع پایه شعاعی (RBF) و سیگموئید. ترجیحی ترین نوع عملکرد هسته RBF است. زیرا موضعی است و در امتداد محور x کامل پاسخ محدودی دارد. توابع هسته محصول اسکالر را بین دو نقطه در یک فضای ویژگی بسیار مناسب برمی گردانند. بنابراین با تعریف مفهوم شباهت ، با کمی هزینه محاسبه حتی در مورد فضاهای با ابعاد بسیار بزرگ. الگوریتم svm از ترفند هسته برای تبدیل نقاط داده و ایجاد مرز تصمیم گیری بهینه استفاده می کند. هسته ها به ما کمک می کنند تا با داده های با ابعاد بالا به روشی بسیار کارآمد برخورد کنیم.

خیر! این کاملاً بستگی به این دارد که شما در واقع چه مشکلی را حل می کنید. اگر داده های شما به صورت خطی جدایی پذیر هستند ، بدون هیچ فکر دیگری ، به دنبال یک هسته خطی بروید. زیرا یک هسته خطی در مقایسه با سایر عملکردهای هسته زمان کمتری را می طلبد. پس انتخاب نوع کرنل بستگی به دیتا و مسئله مورد نظر ما دارد.

- هسته خطی بیشتر برای مشکلات طبقه بندی متن ترجیح داده می شود زیرا در مجموعه داده های بزرگ عملکرد خوبی دارد.
- وقتی اطلاعات اضافی در مورد داده هایی که در دسترس نیستند ، هسته های گاوسی نتایج خوبی دارند.
- هسته Rbf نیز نوعی هسته گوسی است که داده های با ابعاد بالا را فراخوانی می کند و سپس یک جداسازی خطی را برای آن جستجو می کند.
- هسته های چند جمله ای نتایج خوبی برای مشکلاتی که در آن همه داده های آموزش عادی شده است ، می دهند.

2و3-پس از پیش پردازش داده ها با استفاده از دستورات زیر ماژول های مربوط به svm را فراخوانی میکنیم.

```
from sklearn import svm
from sklearn import metrics
from sklearn.metrics import classification_report
```

و همانطور که در تمرین قبل داده های تست و ترین را تقسیم کرده بودیم ، از همان ها استفاده کردیم و کلاس‌بند svm را روی آن ها مدل کردیم. دستور زیر مدل ماشین بردار پشتیبان در حالت خطی است. که همانطور که از خروجی ان ها برمی‌اید هر 3 معیار سنجش برابر 95 درصد است که خب نشانگر خوب بودن مدل روی دیتاست ماست.

```
######Create a # Linear Kernel svm Classifier
clf = svm.SVC(kernel='linear')
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred,average='weighted'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred,average='weighted'))
report =classification_report(y_test, y_pred)
print('report:', report, sep='\n')
```

که در خروجی داریم :

```
Accuracy: 0.9546599496221663
Precision: 0.9547459292411762
Recall: 0.9546599496221663
report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       111
           1       0.94      0.92      0.93        87
           2       0.94      0.95      0.94        98
           3       0.98      0.96      0.97       101

    accuracy                           0.95       397
   macro avg       0.95      0.95      0.95       397
weighted avg       0.95      0.95      0.95       397
```

حال برای svm با استفاده از کرنل RBF مدلسازی را انجام میدهیم. (استاندارد)

```
#Create an rbf Kernel svm Classifier
clf = svm.SVC(kernel='rbf')
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred,average='macro'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred,average='macro'))
report =classification_report(y_test, y_pred)
print('report:', report, sep='\n')
```

و خروجی:

```
Accuracy: 0.21914357682619648
Precision: 0.05478589420654912
Recall: 0.25
report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       111
           1       0.22      1.00      0.36        87
           2       0.00      0.00      0.00        98
           3       0.00      0.00      0.00       101

    accuracy                           0.22       397
   macro avg       0.05      0.25      0.09       397
weighted avg       0.05      0.22      0.08       397
```

که مقادیر پایین دقت ، پرسیژن و ریکال نشان دهنده مناسب نبودن کرنل rbf روی این دیتاست است.

مدل بعدی استفاده از کرنل سیگموییید است.

```
#Create a sigmoid svm Classifier
clf = svm.SVC(kernel='sigmoid')
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred,average='macro'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred,average='macro'))
report =classification_report(y_test, y_pred)
print('report:', report, sep='\n')
```

```
Accuracy: 0.21914357682619648
Precision: 0.05478589420654912
Recall: 0.25
report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       111
           1       0.22      1.00      0.36        87
           2       0.00      0.00      0.00        98
           3       0.00      0.00      0.00       101

    accuracy                           0.22       397
   macro avg       0.05      0.25      0.09       397
weighted avg       0.05      0.22      0.08       397
```

که خروجی فوق نیز عینا مانند کرنل گوسی rbf است و باز هم مدل مناسبی نیست. در ادامه دو مدل rbf با پارامتر های متفوت گامارا نیز مورد بررسی قرار میدهیم. در حالت اول `gamma=1` را در نظر میگیریم.

```
#Create a svm Classifier
clf = svm.SVC(kernel='rbf',gamma=1) # Linear Kernel
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred,average='weighted'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred,average='weighted'))
report =classification_report(y_test, y_pred)
print('report:', report, sep='\n')
```

که خروجی آن فقط برای سنجه Precision با حالت gamma=1 متفاوت است.

```
Accuracy: 0.21914357682619648
Precision: 0.04802390726417908
Recall: 0.21914357682619648
report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       111
           1       0.22      1.00      0.36        87
           2       0.00      0.00      0.00        98
           3       0.00      0.00      0.00       101

    accuracy                           0.22       397
   macro avg       0.05      0.25      0.09       397
weighted avg       0.05      0.22      0.08       397
```

اکنون مدل rbf با پارامتر gamma=12 بررسی میکنیم.

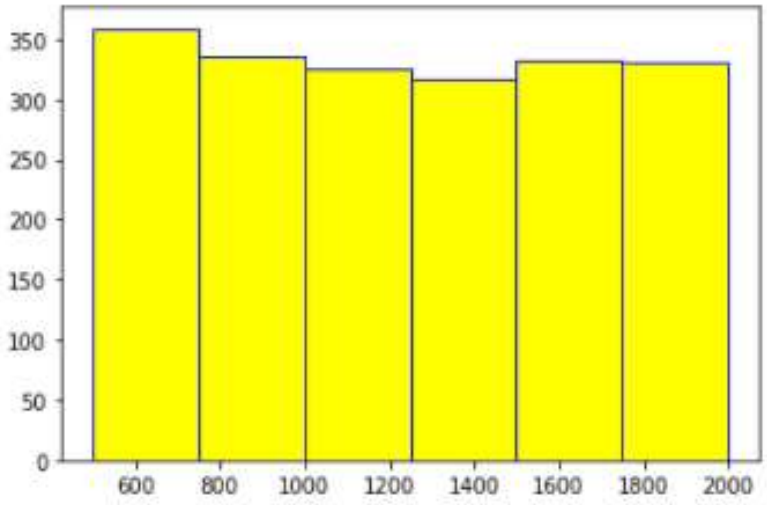
```
#Create a svm Classifier
clf = svm.SVC(kernel='rbf',gamma=12)
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred,average='macro'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred,average='macro'))
report =classification_report(y_test, y_pred)
print('report:', report, sep='\n')
```

```
Accuracy: 0.21914357682619648
Precision: 0.0547858942065491 2
Recall: 0.25
report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       111
           1       0.22      1.00      0.36        87
           2       0.00      0.00      0.00        98
           3       0.00      0.00      0.00       101

    accuracy                           0.22       397
   macro avg       0.05      0.25      0.09       397
weighted avg       0.05      0.22      0.08       397
```

که باز هم مدل ما مناسب نیست . پس تاکنون تنها مدل خطی کرنل قابل پذیرش است.

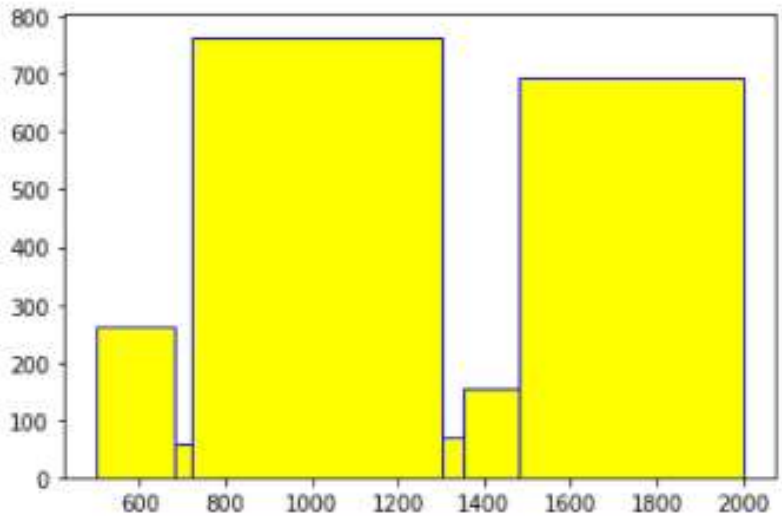
5-الف) برای استفاده از روش بین بندی دوحالت بین با اندازه های مساوی و نامساوی را درنظر میگیریم. کد زیر مربوط به بین های مساوی است که همانطور میبینید در نقاط نزدیک به مینیمم قدرت باتری چگالی بیستری دارند.

```
plt.hist(tdata['battery_power'], bins=6, color="yellow", edgecolor="blue")
```



و حالا بین های نامساوی که به طور تصادفی سایز بندی شده اند:

```
plt.hist(tdata['battery_power'], bins=[501, 680, 720, 1300, 1350,1480, 2000], co
lor="yellow", edgecolor="blue")
```



که برخلاف حالت مساوی در بین های ویط فرکانس بیشتری دارد. و این یعنی با بین بندی در حالت بین های مساوی ما دچار اشتباه میشویم و اطلاعات درستی به ما نمیدهد.

ب) یک one hot encoding به شما اجازه می دهد تا نمایانگر داده های طبقه ای رساتر باشد. بسیاری از الگوریتم های یادگیری ماشین نمی توانند مستقیماً با داده های دسته ای کار کنند. دسته ها باید به اعداد تبدیل شوند. این مورد برای متغیرهای ورودی و خروجی که طبقه بندی شده اند مورد نیاز است. ما می توانیم از یک رمزگذاری صحیح به طور مستقیم استفاده کنیم ، در صورت لزوم ، مجدداً تغییر شکل دهیم. این ممکن است برای مشکلاتی که یک رابطه ترتیبی طبیعی بین دسته ها وجود دارد ، مفید باشد و به نوبه خود مقادیر صحیح ، مانند برچسب های دما "سرد" ، گرم "و" گرم "باشد. ممکن است مشکلاتی وجود داشته باشد که رابطه ترتیبی وجود نداشته باشد و اجازه دادن به نمایندگی برای هر نوع رابطه ای ممکن است برای یادگیری حل مسئله مضر باشد. یک مثال ممکن است برچسب های "سگ" و "گربه" باشد.در این موارد ، ما می خواهیم به شبکه قدرت بیان بیشتری بدهیم تا برای هر مقدار برچسب احتمالی ، یک عدد مشابه را بیاموزد. این می تواند به سهولت در مدل سازی شبکه برای مشکل کمک کند. هنگامی که از یک رمزگذاری داغ برای متغیر خروجی استفاده می شود ، ممکن است مجموعه ای از پیش بینی های دقیق تر از یک برچسب واحد را ارائه دهد.

ج) برای کاهش یا حذف انحراف در توزیع داده های خود و طبیعی تر کردن آن (توزیع A.K.A Gaussian) ، می توانیم از تغییر شکل ورود به سیستم بر روی ویژگی های ورودی خود (X) استفاده کنیم. ما معمولاً توزیع های دنباله دار سنگین را در داده های دنیای واقعی مشاهده می کنیم که در آن مقادیر راست انحراف دارند (مقادیر بزرگتر در توزیع) و چپ انحراف دارند (مقادیر کوچکتر توزیع بیشتر). الگوریتم ها می توانند نسبت به چنین توزیع مقادیری حساس باشند و اگر دامنه به درستی عادی نشوند ، عملکرد متفاوتی را دارند. معمول است که یک تحول لگاریتمی بر روی داده ها اعمال می شود تا مقادیر بسیار بزرگ و بسیار کوچک بر عملکرد الگوریتم یادگیری تأثیر منفی نگذارد. تبدیل ورود به سیستم دامنه مقادیر ناشی از پرت ها را کاهش می دهد. اما مهم است که بخاطر داشته باشید که به محض اینکه تبدیل log انجام شد ، مشاهده داده ها در شکل خام آن دیگر همان معنای اصلی را نخواهند داشت ، همانطور که Log داده را تبدیل می کند.سوال بعدی این است: وقتی رگرسیون خطی انجام می دهیم و ضریب X می گیریم (متغیر مستقل) چگونه متغیرهای مستقل تبدیل شده (X) log را تفسیر می کنیم (اهمیت ویژگی). برای متغیر مستقل (X) ضریب را بر 100 تقسیم کنید. این به ما می گوید که 1٪ افزایش در متغیر مستقل متغیر وابسته را بر (ضریب / 100) واحد افزایش می دهد (یا کاهش می دهد).

در تبدیل نمایی هم همین اتفاق می‌افتد. بطور خلاصه ما به دنبال خطی کردن داده ها هستیم.

د) برای ساخت فیچر جدیدی به نام مساحت یا space کافی است طول و عرض موبایل هارا در هم ضرب نماییم. پس از اینکار ستون جدیدی به نام space به دیتاست ما اضافه خواهد شد. کد زیر را ببینید:

```
tdata['space']= tdata['sc_h']*tdata['sc_w']
tdata.head()
```

فیچر جدید در کادر قرمز رنگ قابل مشاهده است.

Out[22]:

	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range	space
	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1	1	63
	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0	2	51
	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0	2	22
	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0	2	128
	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0	1	16

6-با استفاده از تبدیل رمزنگار و پس از اضافه شدن فیچر های جدید کد زیر را نوشتیم:

```
label_encoder = LabelEncoder()
for cols in tdata.columns:
    tdata[cols]=label_encoder.fit_transform(tdata[cols])
```

```
#Create an rbf Kernel svm Classifier
clf = svm.SVC(kernel='rbf')
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision:",metrics.precision_score(y_test, y_pred,average='macro'))
# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall:",metrics.recall_score(y_test, y_pred,average='macro'))
report =classification_report(y_test, y_pred)
print('report:', report, sep='\n')
```

که خروجی زیر را به ما میدهد.

```
Accuracy: 0.23366834170854273
Precision: 0.05841708542713568
Recall: 0.25
report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       106
           1       0.00      0.00      0.00       100
           2       0.00      0.00      0.00        99
           3       0.23      1.00      0.38        93

    accuracy                           0.23       398
   macro avg       0.06      0.25      0.09       398
weighted avg       0.05      0.23      0.09       398
```

7-درخت تصمیم یکی از روش های ناپارامتری رده بندی کردن است که با توجه به نوع متغیر وابسته به دو دسته رده بندی درختی برای متغیر رسته اي و رگرسیون درختی برای متغیر پیوسته تقسیم می شود.رده بندی درختی در راستای روش های نظیر تحلیل ممیزی و رگرسیون لجستیک است.در این روش مجموعه ای از شرط های منطقی به صورت یک الگوریتم با ساختار درختی برای رده بندی یا پیش بینی یک پیامد به کار می رود.یادگیری درخت تصمیم یکی از پرکاربردترین و سودمندترین روشهای استنتاج استقرایی تابع یاد گرفته شده به صورت درخت تصمیم بازنمایی داده میشود که برای افزایش درجه خوانایی آن برای انسان، درخت را به صورت مجموعهای از قوانین اگر - آنگاه در میآورد.

الگوریتم کای

این الگوریتم توسط کاس در سال 1980 برای استفاده در مورد متغیرهای کیفی معرفی شد که می تواند برای متغیرهای کمی گروه بندی شده نیز استفاده شود. در هر گره، می توان بیش از دو تقسیم نیز داشت. در این روش از آماره کای- دو مربوط به آزمون استقلال جداول توافقی استفاده می شود. از بین متغیر های موجود، متغیری که دارای پی_مقدار کوچک تری باشد در مرحله اول برای تقسیمات روی یک گره در نظر گرفته می شود. ضعف این الگوریتم عدم توانایی آن در ایجاد بهینه ترین تقسیمات ممکن بر .اساس متغیر های موجود است

الگوریتم کرت :

این روش که موجب تشکیل یک درخت تصمیم با تقسیمات دوتایی می گردد، توسط بریمن و همکارانش در سال 1984 به طور کامل معرفی شد. این روش برای متغیرهای کمی طراحی گردیده ولی قابل استفاده برای هر نوع متغیری است. بر اساس این الگوریتم نرم

افزار آماری تحت نام کرت نیز ساخته شده است که از شناخته شده ترین برنامه ها است. در این روش و برای متغیرپاسخ کیفی .،شاخص جینی به عنوان معیاری برای انتخاب متغیرهای مناسب به کار می رود

در معرفی مدل درختی با تقسیمات دوتایی می توان از شاخص های دیگری نظیر آنتروپی نیز استفاده نمود. مزیت شاخص جینی نسبت به آنتروپی و شاخص های دیگر، سرعت بالاتر آن درانجام محاسبات است. مدل کرت را می توان به عنوان یکی از شناخته .شده ترین الگوهای رده بندی به منظور تشخیص وپیشگویی در علوم پزشکی بر شمرد

C4,5 و ID3 الگوریتم های:

توسط کوئینلن در سال 1986 معرفی گردید. این الگوریتم برای معرفی و ساخت درخت رده بندی با تقسیمات چندتایی ID3 الگوریتم در هر گره بسیار مناسب است و این الگوریتم برای متغیرهای کیفی طراحی گردید، ولی می توان از آن برای مجموعه ای از متغیرها، چه کیفی و چه عددی استفاده کرد. ملاک تصمیم گیری در این الگوریتم بر اساس شاخص آنتروپی است که به کمک آن کوئینلن در سال ID3 شاخص های نسبت به دست آمده و اطلاعات محاسبه می شود. با توجه به بعضی از ضعف های الگوریتم اریبی کمتری دارد و برای مشاهدات با مقادیر ID3 معرفی نمود. این الگوریتم نسبت به C4.5 1993 آن را اصلاح و تحت الگوریتم .گمشده مناسب است

نتایج سریع، مختصر و مفید و قابل اطمینان این دو الگوریتم،را به عنوان یک روش قابل قبول برای رده بندی مشاهدات تبدیل نموده .که در علوم پزشکی مورد استفاده قرار می گیرند

الگوریتم کوئست.

این الگوریتم در سال 1997 توسط لو و شی برای متغیرهای پاسخ اسمی طراحی شد. درخت رده بندی حاصل از این الگوریتم نظیر آزمون F مدل کرت دارای تقسیمات دوتایی بوده و ملاک تصمیم برای انتخاب متغیرها با استفاده از پی-مقدار مربوط به آماره

تحلیل واریانس برای متغیرهای کمی و پی-مقدار آماره کای- دو مربوط به جداول توافقی برای متغیر های کیفی صورت می پذیرد. این الگوریتم با توجه به این که از پی-مقدار برای تصمیم گیری استفاده می نماید، موجب تشکیل درختی نااریب ازمتغیرها می گردد. .این الگوریتم ضمن حفظ دقت برآورد در مدل کرت از سرعت بالاتری در معرفی یک درخت رده بندی نسبت ،به آن برخوردار است

الگوریتم کرویز.

این الگوریتم در سال 2001 توسط کیم و لو معرفی گردید که می تواند یک درخت رده بندی با تقسیمات چندتایی معرفی نماید.این الگوریتم به خوبی روش هایی نظیر کوئست و کرت عمل می کند ولی با توجه به این که از تقسیمات چند تایی بهره می برد، از سرعتی بالاتری برخودار است و درخت کوچک تری نیز با استفاده از این الگوریتم معرفی می گردد. درخت معرفی شده در این روش نااریب بوده و طوری طراحی گردیده که باوجود مقادیر گمشده برای داده ها نیز، به خوبی عمل نماید.

به طور خلاصه، درخت تصمیم نقشه‌ای از نتایج احتمالی یکسری از انتخاب‌ها یا گزینه‌های مرتبط بهم است به طوری که به یک فرد یا سازمان اجازه می‌دهد تا اقدامات محتمل را از لحاظ هزینه‌ها، احتمالات و مزایا بسنجد. از درخت تصمیم می‌توان یا برای پیشبرد .اهداف و برنامه‌های شخصی و غیررسمی یا ترسیم الگوریتمی که بر اساس ریاضیات بهترین گزینه را پیش‌بینی می‌کند، استفاده کرد

یک درخت تصمیم‌گیری به طور معمول با یک نُود اولیه شروع می‌شود که پس از آن پیامدهای احتمالی به صورت شاخه‌هایی از آن منشعب شده و هر کدام از آن پیامدها به نُودهای دیگری منجر شده که آن‌ها هم به نوبهٔ خود شاخه‌هایی از احتمالات دیگر را ایجاد می‌کنند که این ساختار شاخه‌شاخه سرانجام به نموداری شبیه به یک درخت مبدل می‌شود. در درخت تصمیم‌گیری سه نوع گره :مختلف وجود دارد که عبارتند از

- نُودهای تصادفی

- نُودهای تصمیم‌گیری

- نُودهای پایانی

نُود تصادفی، که توسط یک دایره نشان داده می‌شود، نمایانگر احتمال وقوع یکسری نتایج خاص است، نُود تصمیم‌گیری، که توسط یک مربع نشان داده می‌شود، تصمیمی که می‌توان اتخاذ کرد را نشان می‌دهد و همچنین نُود پایانی نمایانگر پیامد نهایی یک مسیر تصمیم‌گیری خواهد بود.

8- با استفاده از الگوریتم انتروپی کد زیر را نوشتیم ودر نهایت بعد از پیش بینی تارگت و محاسبه دقت به عدد 82درصد رسیدیم. که دقت قابل قبولی است ولی خوب نیست.

```
#Import the DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
#making the model with entropy methode
tree = DecisionTreeClassifier(criterion = 'entropy').fit(X_train,y_train)
#make prediction
prediction = tree.predict(X_test)
#Check the accuracy
print("The prediction accuracy is: ",tree.score(X_test,y_test)*100,"%")
```

```
The prediction accuracy is:  81.90954773869346 %
```

9-درخت تصمیم‌گیری سعی می‌کند به صورت بازگشتی داده‌ها را به قسمی از هم جدا کند که در هر گِره متغیرهای مستقلِ به هم نزدیک شده همسان شوند هر گِره زیر مجموعه ای از داده هاست که به صورت بازگشتی ساخته شده‌است.

حال سؤال اینجاست که کدام بُعد از متغیرهای وابسته و چه آستانه‌ای را باید انتخاب کرد. به زبان ریاضی باید آنθیی را انتخاب کرد که ناخالصی داده را کم کند. ناخالصی برحسب نوع مسئله تعریفی متفاوت خواهد داشت، مثلا اگر مسئله یک دسته‌بندی دوگانه است، ناخالصی می‌تواند آنتراپی داده باشد، کمترین ناخالصی زمانی است که هم$Q_{right}(\theta)$ و هم$Q_{left}(\theta)$ از یک دسته داشته باشند، یعنی در هر کدام از این دو گِرِه دو نوع دسته وجود نداشته باشد. برای رگرسیون این ناخالصی می‌تواند واریانس متغیر وابسته باشد. هدف در اینجا پیدا کردن آن θیی است که ناخالصی را کمینه کند، یعنی .حال همین کار را به صورت بازگشتی برای هم$Q_{right}(\theta)$ و هم$Q_{left}(\theta)$ انجام می‌دهیم. بعضی از گره‌ها را باید به برگ تبدیل کنیم، معیاری که برای تبدیل یک گره به برگ از آن استفاده می‌کنیم می‌تواند مقداری حداقلی برای تعداد داده در یک گره و یا عمق درخت باشد به قسمی که اگر با دو نیم کردن گِرِه یکی از معیارها عوض شود، گِرِه را به دو نیم نکرده آن را تبدیل به یک برگ میکنیم. معمولاً این دو پارامتر باعث تنظیم مدل (Regularization) می‌شونددر ابتدای کار گره شامل تمام داده‌ها

می‌شود .

10- هرس روشی در یادگیری ماشین است که باعث کاهش اندازه درخت‌های تصمیم‌گیری با از بین بردن بخشهایی از درخت است که قدرت کمی برای طبقه‌بندی نمونه‌ها دارند. هرس باعث کاهش پیچیدگی طبقه‌بندی کنندهٔ نهایی می‌شود و از این رو باعث بهبود دقت پیش بینی و کاهش بیش برازش می‌شود.

12-با استفاده از پکیج سایکیت لرن ما یک رندوم فارست را مدل کردیم.به صورت زیر

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X_train,y_train)
```

با خروجی زیر

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=2, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

و پیشبینی تارگت را با دستور زیر انجام دادیم:

```
print(clf.predict(X_test))
```

که پیش بینی تارگت ها را میتوان مشاهده کرد:

```
[0 0 0 3 0 0 1 2 3 0 2 1 0 2 0 0 2 3 0 2 3 0 2 1 3 0 0 0 0 2 3 3 1 0 2 3 1
 3 0 1 3 2 2 1 1 3 0 3 1 0 1 3 1 0 1 0 3 0 2 2 1 2 2 1 0 1 1 3 3 0 3 0 1 1
 2 1 1 0 3 3 0 0 2 3 3 0 0 0 3 1 3 1 1 0 0 1 0 3 0 0 3 2 0 3 1 2 2 0 3 3 0
 1 0 2 2 0 2 1 1 1 1 2 1 3 1 0 0 0 1 1 1 0 1 3 2 1 0 3 1 0 2 1 3 2 2 2 3 2
 1 1 1 0 0 3 3 0 3 1 1 1 3 2 3 1 3 0 1 3 3 2 1 0 1 2 1 3 2 3 2 0 0 0 1 0 1
 0 1 2 1 1 0 2 3 0 0 2 1 3 1 1 0 1 0 1 1 1 3 2 2 0 3 0 0 2 3 1 1 0 0 3 1 1
 1 1 0 3 2 2 3 0 3 1 2 3 3 0 2 0 3 1 3 0 1 1 1 1 2 1 2 1 0 0 2 2 3 1 1 2 0
 1 1 3 2 3 1 3 2 1 1 0 0 3 0 3 2 3 3 1 0 2 3 2 3 3 2 0 0 1 1 0 1 0 1 0 3 2
 3 0 1 2 1 3 2 2 2 3 0 1 3 1 0 3 3 0 2 0 0 0 1 0 0 2 1 0 0 0 0 3 3 0 2 1 1
 1 3 3 0 1 0 1 1 1 0 1 0 0 0 3 0 2 2 0 0 0 0 3 3 3 3 1 1 0 1 0 0 3 0 1 3 0
 2 0 0 2 3 1 2 3 2 2 1 1 0 3 0 1 0 2 1 1 2 3 3 0 0 3 2 2]
```

صحت مدل را به طور زیر بررسی کردیم که نتیجه خیلی بد بود!!!!

```
clf.score(X_test,y_test )
```

که دقت مدل عدد زیر را نشان داد

```
0.678391959798995
```

این یعنی مدل ما زیاد جالب نیست.

13-آموزش آنها سریع است ، دقت مناسبی دارند و بسیار قابل توضیح هستند. شما می توانید به معنای واقعی کلمه ببینید که چرا مدل فقط آنچه را که گفته فقط با نگاه به درخت گفته است. این مثلاً در برابر یک شبکه عصبی است ، که هیچ توضیحی درباره آنچه انجام می دهد به شما نمی دهد.

15-Random Forest یک الگوریتم محبوب و موثر برای یادگیری ماشین در گروه است.این به طور گسترده ای برای طبقه بندی و رگرسیون مشکلات مدل سازی پیش بینی با ساختار داده (جدول) مجموعه داده ها استفاده می شود ، به عنوان مثال. داده ها همانطور که در صفحه گسترده یا جدول پایگاه داده به نظر می رسند.از Random Forest می توان برای پیش بینی سری های زمانی نیز استفاده کرد ، اگرچه این امر مستلزم آن است که ابتدا مجموعه داده های سری زمانی به یک مسئله یادگیری تحت نظارت تبدیل شود. همچنین به استفاده از یک تکنیک ویژه برای ارزیابی مدل موسوم به اعتبار سنجی حرکت به جلو نیاز دارد ، زیرا ارزیابی مدل با استفاده از اعتبار صلیبی k برابر می تواند منجر به نتایج مغرضانه خوشبینانه شود.