گزارش ۷ درس هوش مصنوعی

مقایسه روشهای مختلف طبقه بندی روی یک مسئله benchmark

به قلم:
امیر بابامحمودی

استاد
دکتر مهدی قطعی

خرداد ۱۴۰۰

مقدمه:

در این گزارش قصد داریم دو مدل قدرتمند در یادگیری ماشین را روی یک دیتا ست پیاده سازی کرده و آن ها را با یکدیگر مقایسه کنیم.

دیتاست titanic :

لینک دریافت دیتاست:

https://www.kaggle.com/c/titanic/data

این دیتا ست شامل اطلاعات ۸۹۱ مسافر کشتی تایتانیک میباشد که با استفاده از این اطلاعات میخواهیم ببینیم که کدام یک از مسافران نجات یافته و کدام یک غرق شده اند.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S

از اطلاعات موجود در جدول بالا نیاز میباشد که ستون survived رو به عنوان target جدا کرده و ستون های جنسیت, سن, SibSp که همان همسر و خواهر برادر مربوط به مسافر که سوار کشتی بوده است, Parch که پدر مادر و یا فرزند مسافر که مسافر کشتی باشند و همچنین لوکیشنی که سوار کشتی شدند یعنی embarked را برای train کردن استفاده کنیم.

```
gender = {'male': 1,'female': 2}
titanic.Sex = [gender[item] for item in titanic.Sex]
titanic = titanic[titanic['Embarked'].notna()]
Embark = {'S' : 1 , 'C' : 2 , 'Q' : 3}
titanic.Embarked = [Embark[item] for item in titanic.Embarked]
```

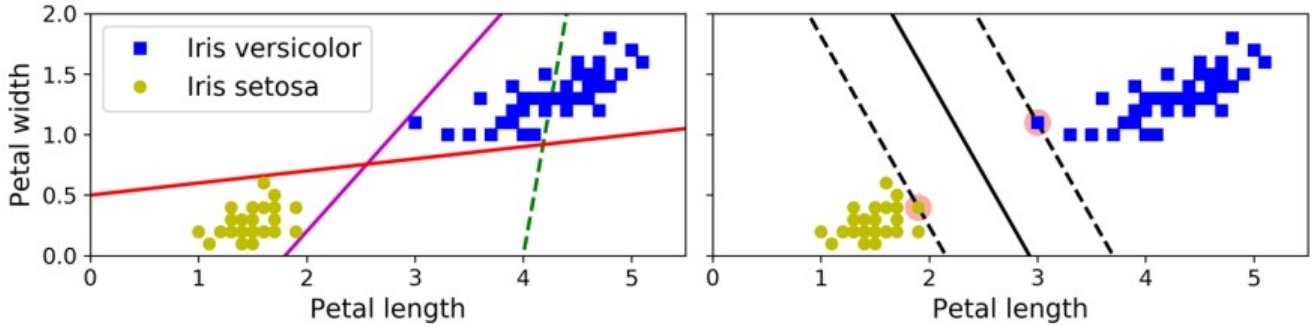
ابتدا ستون های شامل رشته رو به یک عدد کد میکنیم برای مثال جنسیت مرد ۱ و خانم ۲. و همچنین برای Embarked.

حال دو جدول متفاوت ساخته یکی برای داده های ورودی دیگری به عنوان لیبل که نجات یافته اند یا خیر. سپس این دو جدول را به نسبت۷۵ به ۲۵ برای trainset و testset تقسیم میکنیم.که کد آن مانند شکل زیر میشود.

```
titanic_select = titanic[[ "Pclass", "Sex", "Age", "SibSp", "Parch", "Embarked"]].copy()
survived = titanic[["Survived"]]
```

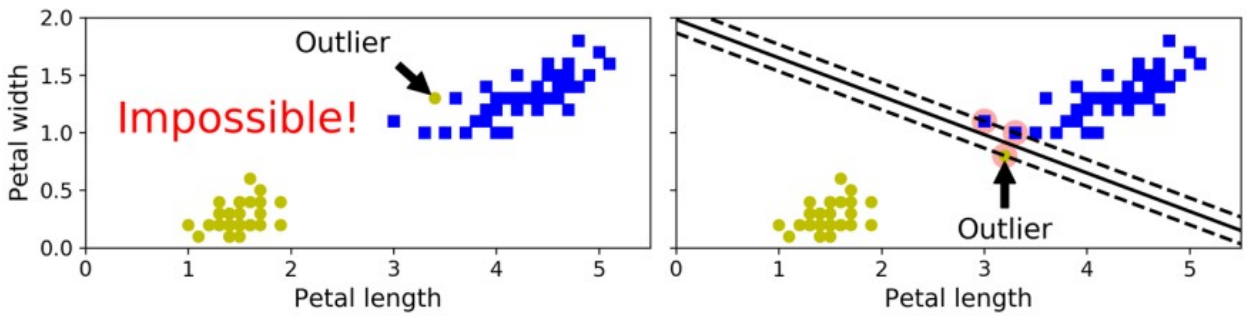
```
x_train = titanic_select[:650]
y_train = survived[:650]
x_test = titanic[650:]
y_test = survived[650:]
```

SVM: اختصاری از support vector machine یک مدل قدرتمند یادگیری ماشین که میتوان با آن طبقه بندی خطی و غیر خطی انجام داد.ایده ی اصلی SVM را میتوانید در تصویر زیر مشاهده کنید. دو نوع کلاس از گل ها میتوانند به راحتی با خطی از یکدیگر جدا بشوند.



همانطور که در سمت راست میبینید خط صاف کشیده شده کاملا دو نوع گل رو از هم جدا کرده.این نکته شایان ذکر است که نمونه های اضافه شده ای که خارج از محدوده ی اصطلاحا خیابان در سمت راست باشند هیچ تاثیری نداشته و داده هایی که روی لبه ی خط چین میباشند موثرند که به آن ها support vectors میگویند.

اگر بخواهیم به صورت hard margin classification کار کنیم مشکلاتی دارد. برای مثال این نوع طبقه بندی تنها برای داده هایی که به صورت خطی جداپذیر باشند به کار میرود و اینکه هر داده باید حتما خارج از محدوده ی خیابان باشد وگرنه نمیتوان آنرا اعمال کرد. مانند شکل زیر که شکل سمت چپ مشکل ساز است

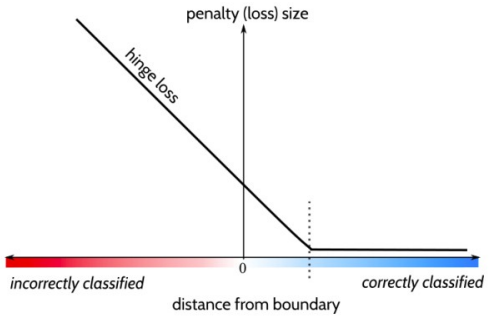


حال برای جلوگیری از این مشکل از soft margin classification استفاده میکنیم که حساسیت کمتر داشته و میتواند margin vaiolation (داده در محدوده ی خیابان) داشته باشد. حال به کد با این روش میپردازیم.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
from sklearn import preprocessing
from sklearn.metrics import classification_report
svm_clf = Pipeline([
("scaler", StandardScaler()),
("linear_svc", LinearSVC(C=1, loss="hinge")),])
svm_clf.fit(x_train, y_train)
scalar_train= preprocessing.StandardScaler().fit(x_train)
scalar_test= preprocessing.StandardScaler().fit(x_test)
x_train_scale = scalar_train.transform(x_train)
x_test_scale = scalar_test.transform(x_test)
svm_clf.fit(x_train_scale, y_train)
prediction = svm_clf.predict(x_test_scale)
print(classification_report(prediction , y_test))
```

```
              precision    recall  f1-score   support

           0       0.87      0.80      0.84       121
           1       0.66      0.77      0.71        61

    accuracy                           0.79       182
   macro avg       0.77      0.79      0.77       182
weighted avg       0.80      0.79      0.79       182
```

همانطور که در کد مشاهده میکنید ابتدا یک pipline طراحی کرده که در آن داده ها را به صورتی scale کرده که دارای توزیع نرمال شوند و با تابع hinge loss سعی در بهتر کردن پارامتر ها جهت کاهش hinge loss کرده که شکل نمودار و فرمول این cost function را در زیر میبینید.



$$l = max(0, 1 - y^i(x^i - b))$$

$$l = \left\{ \begin{array}{l} 0 \quad if\ y \cdot (w \cdot x) \geq 1 \\ 1 - y \cdot (w \cdot x)\ otherwise \end{array} \right\}$$

سپس داده هارا با این مدل train کرده و خروجی را با سه نوع ارزیابی precision, recall و f1-score بررسی میکنیم که در ادامه ی گزارش به شرح این سه نوع ارزیابی و مقایسه با مدل های دیگر خواهیم پرداخت.

Decision Tree:

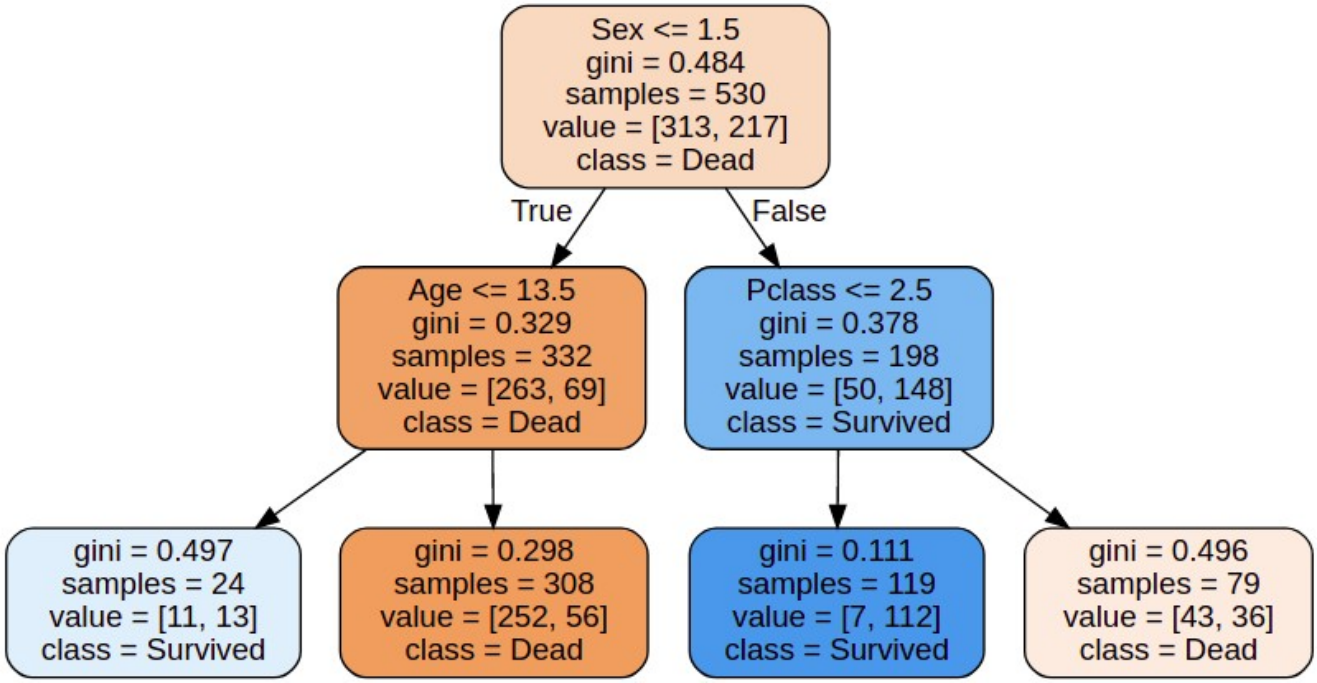
شرح این بخش را پس از اجرای کد توضیح میدهیم:

```
from sklearn.tree import DecisionTreeClassifier
tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(x_train, y_train)
prediction_dt = tree_clf.predict(x_test)
print(classification_report(prediction_dt , y_test))
```

```
              precision    recall  f1-score   support

           0       0.94      0.79      0.86       131
           1       0.62      0.86      0.72        51

    accuracy                           0.81       182
   macro avg       0.78      0.83      0.79       182
weighted avg       0.85      0.81      0.82       182
```

```
from sklearn.tree import export_graphviz
from graphviz import Source

target_name = ["Dead" , "Survived"]
feature_names = ["Pclass", "Sex", "Age", "SibSp", "Parch", "Embarked"]
export_graphviz(
tree_clf,
out_file="graph.dot",
feature_names=feature_names,
class_names=target_name,
rounded=True,
filled=True
)
Source.from_file("graph.dot")
```

در ابتدا کلاس مرتبط به آن را ساخته و داده هارا با آن train میکنیم و با testset معیار هارا اندازه گیری میکنیم. سپس میتوانیم به وسیله ی کتابخانه ی sklearn و متد export_graphviz آن را نحوه ی پیش بینی این مدل را بررسی کنیم که تصویر آن به شکل زیر است.



فرض کنید با توجه به اطلاعات یک فرد میخواهید زنده یا مرده بودن او را بررسی کنید. در ابتدا در ارتفاع صفر و node ریشه را میبینیم که اگر جنسیت مرد باشد که آن را با عدد ۱ کد کردیم به سمت چپ رفته و سوال دیگری میپرسد. اگر سن او کمتر از 13.5 سال باشد احتمال زنده بودن او بیشتر است. همانطور که میبینید sample تعداد افراد با یک ویژگی خاص را میشمارد. برای مثال ۱۱۹ نفر خانم با درجه بیلیط ۲ به پایین وجود دارد. و اما gini در اصل میزان ناخالصی یک کلاس را میسنجد. به این صورت که اگر صفر باشد یعنی تمام افراد آن کلاس نتیجه ی یکسان با آن کلاس داشتند برای مثال همگی زنده ماندند. برای مثلا حدود ۳۰ درصد از مرد های بالا 13.5 سال زنده ماندند که یعنی ۵۶ نفر از آن هازیرا gini عدد حدودا 0.3 را نشان میدهد در صورتی که در کلاس کسانی که مردند هستیم. در کل هم میبینیم که منطقی بنظر میرسد خروجی زیرا در کشتی تایتانیک اولویت با زن ها کودکان و پولداران برای نجات یافتن بود.

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

فرمول محاسبه ی gini هم به صورت مقابل میباشد:

که pi,k نرخ افراد کلاس k در کل داده های نود i میباشد.

:MLP

اختصاری از Multilayer Perceptron شامل لایه هایی از نورون ها که با گرفتن ورودی و انجام دادن پردازش هایی روی آن پارامتر های مرتبط با هر لایه رو حساب کرده و در نهایت در لایه ی آخر به تعداد کلاس ها نورون وجود دارد که احتمال تعلق هر دیتا به این کلاس ها را میسنجد. از آنجایی که دیتای انتخابی در این گزارش بسیار سبک بوده و کم میباشد از تنها دو hidden layer استفاده شده است.

```
import tensorflow as tf
from tensorflow import keras
model = keras.models.Sequential([
keras.layers.InputLayer(input_shape = 6),
keras.layers.Dense(18, activation="relu"),
keras.layers.Dense(10, activation="relu"),
keras.layers.Dense(2, activation="sigmoid")])
model.compile(loss="sparse_categorical_crossentropy",
optimizer="sgd",
metrics=["accuracy"])
history = model.fit(x_train_scale , y_train , epochs=20 )
predict_mlp = model.predict_classes(x_test_scale)
print(classification_report(predict_mlp , y_test))
```

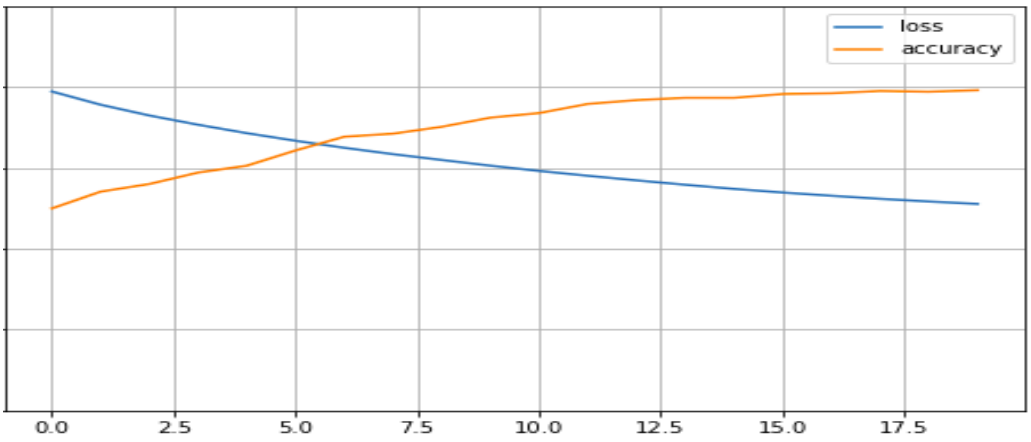
```
Epoch 1/20
17/17 [==============================] - 0s 814us/step - loss: 0.6554 - accuracy: 0.7113
Epoch 2/20
17/17 [==============================] - 0s 966us/step - loss: 0.6300 - accuracy: 0.7283
Epoch 3/20
17/17 [==============================] - 0s 1ms/step - loss: 0.6095 - accuracy: 0.7434
Epoch 4/20
17/17 [==============================] - 0s 981us/step - loss: 0.5938 - accuracy: 0.7491
Epoch 5/20
17/17 [==============================] - 0s 989us/step - loss: 0.5798 - accuracy: 0.7642
Epoch 6/20
17/17 [==============================] - 0s 961us/step - loss: 0.5677 - accuracy: 0.7698
```

در لایه اول ابعاد ورودی که شامل تنها یک ارایه یک بعدی شامل ۶ feature میباشد را میدهیم و در لایه های hidden از تابع relu استفاده شده و در نهایت در لایه آخر که شامل دو کلاس زنده ماند یا مردن است از sigmoid استفاده میکنم.

	precision	recall	f1-score	support
0	0.90	0.83	0.86	121
1	0.70	0.82	0.76	61
accuracy			0.82	182
macro avg	0.80	0.82	0.81	182
weighted avg	0.83	0.82	0.83	182

که خروجی برای معیار ها به این صورت است.

```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
plt.show()
```



همانطورکه در نمودار رسم شده میبینید با طی کردن هر epoch دقت افزایش یافته و loss function در حال کاهش میباشد.

حال به شرح معیار هایی که برای مقایسه سه مدل پیاده سازی شده استفاده شده است میپردازیم.

:Precision

$$\text{precision} = \frac{TP}{TP + FP}$$

که TP تعداد داده های مثبتی که درست حدس زده شده اند و FP داده هایی که به اشتباه درست حدس زده شده اند میباشد.(یعنی زنده تشخیص داده شده اند)

:recall

$$\text{recall} = \frac{TP}{TP + FN}$$

FN تعداد داده های منفی ای که به اشتباه منفی حدس زده شده اند.(در اینجا منفی یعنی مرده ها)

F1-Score:

ترکیبی از دو فرمول ذکر شده در بالا میباشد که به آن میانگین موزون(harmonic mean) هم گفته میشود.و دلیل این نام گذاری این میباشد که به مقادیر کمتر ارزش بیشتری میدهد.که این بدان معنی میباشد که این معیار زمانی مقدارش زیاد میشود که هر دو معیار دیگر مقدار زیادی داشته باشند.

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

حال به مقایسه خروجی هر سه روش با این سه معیار میپردازیم:

مدل/معیار سنجش	recall	precision	F1-Score
SVM	0.79	0.8	0.79
Decision Tree	0.81	0.85	0.82
MLP	0.82	0.83	0.83

دلیل اینکه درصد پیشبینی های testset به میزان خوبی مطلوب نشد را میتوان کم بودن تعداد داده ها که حدود ۷۰۰ داده موجود بود گذاشت. و دلایل دیگری نظیر بهینه انتخاب نکردن درصد مورد نظر برای trainset , testset دانست. همچنین معیار هایی برای مقدار دهی اولیه به یک سری پارامتر وجود دارد که رندوم میباشند و با هر بار ران کردن مدل ها جواب ها به میزان کمی متفاوت خواهند بود و به طور کلی یک سری hyperparametr ها وجود دارند که با آزمایش ها متعدد میتوان در مقدار دهی آن ها بهینه تر عمل کرد.

منابع:

۱ـ https://towardsdatascience.com/a-definitive-explanation-to-hinge-loss-for-support-vector-machines-ab6d8d3178f1

۲ـ hands-on machine learning with scikit-learn and tensorflow aurelien geron

لینک گیتهاب جهت دسترسی به کد:
https://github.com/amirbabamahmoudi/AI-projects/tree/main/classification_compare_models