Machine Learning
VG Rapporten
Ronja Larsson
2023-04-04

Introduktion

Bakgrund

Vi kan lösa många komplexa problem med hjälp av maskininlärning. Maskininlärning innebär att man programmerar datorer så att de kan lära sig själva från data som vi har samlat och matat in. Syftet med maskininlärningsprojekt är att göra prediktioner. När en maskininlärningsmodell har lärt sig sambanden som finns i träningsdatan går man vidare till utvärderingen av hur väl modellen kan prediktera genom att testa modellen på valideringsdatan och testdatan. Man brukar modellera flera olika modeller och sedan väljer man den modellen som ger mest korrekta prediktioner baserat på utvalda mättningsmått.

MNIST är ett känt dataset som innehåller handskrivna siffror som många använder för att utvärdera och testa olika maskininlärningsmodeller. När man studerar maskininlärning så kommer man, förr eller senare, att få jobba med detta dataset. MNIST är datasettet som vi kommer att använda för vår modellering i detta maskininlärningsprojekt.

Frågeställning

Då syftet med maskininlärning är att göra prediktioner, och gärna så korrekt som möjligt, så är frågeställningen som vi kommer att behandla i detta projekt följande:

- Hur väl kan vi träna en maskininlärningsmodell för att känna igen handskrivna siffror och ge ett korrekt svar på vilken siffra som visas för maskininlärningsmodellen?

Frågeställningen undersöker prediktionsförmåga som kommer att mätas i accuracy score och F1 score från Confusion Matrix eftersom jag inte kommer att specificera en user case där modellen ska användas.

EDA

MNIST är ett dataset som består av handskrivna siffror. Det består av 70 000 observationer och 784 features. En observation motsvarar en rad och en feature motsvarar 784 kolumner. Varje observation representerar en bild på en handskriven siffra och varje kolumn i en observation representerar en pixel som bilden består av. Bilderna i detta dataset är i en gråskala på 28x28 pixlar som är då de 784 kolumnerna.

MNIST passar bra för att testa maskininlärningsmodeller på verkliga problem utan att spendera för mycket tid på att tvätta data eftersom datan är redan tvättad och formaterad. Jag valde att göra minimalt EDA eftersom detta dataset, som bekant, är redan tvättad och att jag inte vill själv få en uppfattning på hur datan ser ut när jag bygger ML modeller.

Jag delade datan in i X och y variabler och dubbelkollade att formen på datan ser korrekt ut:

```
In [7]: # Saving the data and target in X och y variables.
        X = mnist["data"]
        y = mnist["target"].astype(np.uint8)

In [8]: # Checking the shape of X and y.
        print(X.shape) # X contains 70000 observations with 784 features.
        print(y.shape) # y contains 70000 correct answers: labels.

        (70000, 784)
        (70000,)
```

För nyfikenhetensskulle vill jag kolla vilket nummer den 1124:e bilden föreställer och bilden visar siffran 2.

Då det tar lång tid att träna maskininlärningsmodeller har jag valt att ta endast 6 000 observationer som träningsdata och 1 000 observationer som valideringsdata på träningsdata. Både träningsdata och valideringsdata användes för att träna samtliga modeller. Senare har jag tagit ytterligare 1 000 observationer som testdata. Testdata användes på de tre ensemble-modeller (VOT, RF, RF med GS) för att se hur bra de generaliserar. Slutligen tog jag efterföljande 10 000 som andra grupp av testdata för att se hur väl den modellen som jag till slut valde generaliserar.

Machine Learning
VG Rapporten
Ronja Larsson
2023-04-04

Metod och Modeller

Jag valde att utveckla min modell genom två steg. I första steget tränade jag tre basmodeller (SVM, SGDClassifier, KNeighborsClassifier) med hjälp av GridSearch för att ta fram kombinationen av hyperparameter som ger bästa prediktioner. Dessa tre basmodeller ska användas i en VotingClassifier modell. I steg två tränade jag tre modeller, en VotingClassifier, en RandomForest utan GridSearch och en RandomForest med GridSearch.

VotingClassifier är en ensemble learning metod som kombinerar prediktioner från flera olika modeller för att förbättra modellens noggrannhet. Det är en enkel men kraftfull teknik som ofta kan förbättra prediktionsförmåga och uppnå bättre total noggrannhet. Jag har byggt en VotingClassifier som innehåller SVM, SGDClassifier och KNeighborsClassifier.

RandomForest är en ensemble-modell som kombinerar flera beslutsträd som tränas på olika delar av träningsdatan. Denna modell är robust mot overfitting och kan hantera stora datamängder. Jag har valt att modellera med RandomForest eftersom den är effektiv för bild- och textclassifiering. Jag byggde två versioner av RandomForest, en utan GridSearch och en med GridSearch. I GridSearch för min RandomForest modell har jag justerat hyperparameter som n_estimators, max_depth, min_samples_split och min_samples_leaf. Jag vill jämföra hur bra en RandomForest modell presterar med och utan hjälp från GridSearch.

Slutligen stod valen mellan VotingClassifier och de två RandomForest modeller. För att utvärdera dessa tre modeller har jag valt att använda mättningsmåtten accuracy score och F1 score. Accuracy score mäter den totala korrektheten hos prediktionerna, oavsett klass. Den hänvisar till förhållandet mellan antalet korrekta prediktioner (sanna positiva och sanna negativa) över det totala antalet prediktioner. Med andra ord mäter det procentandelen av alla observationer som är korrekt klassificerade av modellen. F1 score är ett mått på en modells noggrannhet som tar hänsyn till både precision och recall. Det är det harmoniska medelvärdet av precision och recall, med ett värde som varierar mellan 0 och 1. En hög F1 score indikerar både hög precision (mäter andelen av de predikterade positiva instanserna som verkligen var positiva) och hög recall (andelen av alla relevanta instanser som modellen identifierade korrekt), vilket innebär att modellen gör korrekta positiva prediktioner och inte missar några relevanta resultat.

Då jag inte har tänkt ut en specifik user case för denna modellering så anser jag att det är rimligt att titta på modellens övergripande noggrannhet och försöker prediktera så korrekt som möjligt. I detta fall blir accuracy score och F1 score de självklara valen som mättningsmått.

Resultat och Analys

Resultat

Resultat på *valideringsdata* av 1 000 observationer för VotingClassifier och RandomForest visas nedan. Accuracy score och F1 score på VOT och SVM är samma, och båda versionerna av RF ger samma scores för accuracy och F1.

Voting	Accuracy	F1
SVM(bas)	93%	0.93
SGD(bas)	91%	0.91
KNN(bas)	90%	0.89
VOT	93%	0.93

RandomForest	Accuracy	F1
RF	95%	0.95
RF(GS)	95%	0.95

Resultat på *testdata* av 1 000 observationer för de tre ensemble-modellerna visas nedan. Vi kan se att VOT behåller samma scores medan båda versioner av RF har fått mycket lägre score.

	Accuracy	F1
VOT	93%	0.93
RF	78%	0.75
RF(GS)	80%	0.79

Eftersom VOT behåller samma score på både validerings- och testdata bestämde jag att köra en till testdata som är större än tidigare. Jag tog ytterligare 10 000 observationer som den andra testdatan och matade datan in på VOT för att se hur väl den predikterar på en större testdata. Resultatet visas nedan. Vi kan se att accuracy score är 92% och F1 score är 0.92, vilket inte är så långt från tidigare scores (93% accuracy och 0.93 F1).

```
In [33]: # Chosen model: Voting
y_test_2_pred = vot.predict(X_test_2_scaled)

print('Accuracy score: ', accuracy_score(y_test_2, y_test_2_pred, normalize = True))
print('F1 score: ', f1_score(y_test_2, y_test_2_pred, average = 'weighted'))
print(classification_report(y_test_2, y_test_2_pred))

display_confusion_matrix(y_test_2, y_test_2_pred)
```

```
Accuracy score:  0.9234
F1 score:  0.9233203939901891
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       988
           1       0.96      0.98      0.97      1145
           2       0.91      0.92      0.92       952
           3       0.93      0.89      0.91      1072
           4       0.92      0.93      0.92       948
           5       0.93      0.91      0.92       885
           6       0.94      0.95      0.95       973
           7       0.95      0.91      0.93      1047
           8       0.90      0.86      0.88       979
           9       0.85      0.91      0.88      1011

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000
```

Analys

I det första steget verkade RandomForest ge bättre resultat än VotingClassifier på valideringsdata, men när jag testade modellerna på testdatan visade sig att RandomForest modellerna ge mycket lägre scores. Detta tyder på att modellerna överanpassar till träningsdatan och memoriserar träningsdata utan att lära sig mönster. Då RandomForest genaliserar dåligt bestämde jag att testa VotingClassifier som fick samma score på båda valideringss- och testdata (93% på accuracy scores och 0.93 på F1).

RF	Accuracy	F1
Validation	95%	0.95
Test	78%	0.75

RF(GS)	Accuracy	F1
Validation	95%	0.95
Test	80%	0.79

Efter jag har testat VOT-modellen två gånger så ser vi att accuracy scores på alla tre prediktioner (validation, test, test_2) är ungefär på samma range. Detta visar på att modellen är pålitlig och generaliserar bra utanför tränings- och valideringsdata. En förklaring till varför accuracy och F1 är något lägre kan vara för att test_2 består av tio gånger fler antal observationer jämfört med valideringsdata (1 000 observationer) och testdata (1 000 observationer).

VOT	Accuracy	F1
Validation	93%	0.93
Test	93%	0.93
Test_2	92%	0.92

Anledningen till att VOT generaliserar bättre än RF-modellerna kan vara på grund av dess förmåga att kombinera flera modeller för att ta hänsyn till olika styrkor och svagheter hos varje basmodell. VOT kombinerar SVM, SGD och KNN och aggregerar deras resultat för att få en mer robust klassificeringsprestation. I min VOT-modell har jag valt att tillämpat soft voting eftersom den tillåter de tre basmodellerna att bidra med sina predikterade sannolikheter i stället för bara deras klassetiketter. Detta kan hjälpa till att bättre fånga osäkerheten i prediktionerna och detta kan vara anledning att modellen generaliserar bra även på en större testdata.

Slutsats

Efter mina tester på VOT med tillfredsställande resultat har jag valt VOT som min modell för att känna igen handskrivna siffror. Då jag inte har specificerat en user case, anser jag att VOT-modellen presterar tillräckligt bra i nuläget. Innan jag bevarar på frågeställning för detta projekt vill jag upprepa den här igen:

- Hur väl kan vi träna en maskininlärningsmodell för att känna igen handskrivna siffror och ge ett korrekt svar på vilken siffra som visas för maskininlärningsmodellen?

Vi kan träna en VotingClassifier som predikterar med ca 93% på accuracy score samt 0.93 på F1 score, det vill säga av 1000 siffror kan modellen korrekt känna igen 930 av siffrorna och F1 på 0.93 visar en ganska hög precision och recall.

Potentiell vidareutveckling

Beroende på vad man ska använda modellen till så kan man behöva förbättra modellen ytterligare. Med hjälp av Confusion Matrix kan man också se att modellen prediketar sämre på vissa siffror (7 och 9), så man skulle kunna vidareutveckla modellen så att den kan predikera lika bra på olika siffror. Man kan också använda neurala nätverk för att utveckla en ny modell som kan eventuellt öka prediktionsförmåga ytterligare.

Kort Redogörelse

1. *Utmaningar du haft under arbetet samt hur du hanterat dem.*

   Utmaningen låg i att uppgiften är ganska fri i formen och man kan vara väldigt kreativ med hur man ska utforska datasettet. Jag känner att jag fortfarande saknar intuitionen för att kunna modellera på det bästa sättet. Det är många aspekter som jag inte kan se ännu som en erfaren Data Scientist skulle se på en gång. Jag satt fast ett tag i att testa flera modeller med olika hyperparameter utan att kunna fatta ett beslut på vilken modell jag ska välja och varför.

   Jag ville också få högsta accuracy score på modellerna som möjligt. Jag visste inte heller vad räknades som en bra score eller inte, så jag gjorde flera modeller och sedan plocka bort de modellerna med mindre scores. Det är väldigt många olika modeller man skulle kunna testa och en del av mig ville testa alla vi gick igenom i kursen bara för att se vad som händer när olika modeller används. Detta var dock för tidskrävande.

   För att hitta en mellangrund, bestämde jag för att strukturera min modellering i två steg där jag fick leka med flera modeller som basmodeller som jag använde på en av ensemble-modellerna. Anledningen till varför jag ville använda ensemble-modellerna som VotingClassifier och RandomForest var just för att jag ville försöka få en högre score och att jag fick använda några av de första modellerna jag skapade när jag satt fast med för många modeller.

   Jag har hanterat dessa problem genom att nå ut till läraren och klasskamrater och be dem att titta på min kod och ge feedback. De kanske ser något jag har missat och det är alltid bra att få input när man själv har suttit i problemet för länge.

2. *Vilket betyg du anser att du skall ha och varför.*

   Jag anser att jag ska ha VG. Jag har på ett fördjupat sätt tillämpat maskininlärningsmodeller och metoder genom att använda VotingClassifier där jag också använt GridSearch för att välja ut en modell som ger bäst resultat. Jag har visat att jag kan på ett kreativt och självständigt sätt skapa en modell och visualiserar mina resultat.

3. *Tips du hade ”gett till dig själv” i början av kursen nu när du slutfört den.*

   Att på förhand börja göra flera ML kurserna i DataCamp – många koncept som vi har pratat om i kursen kan man också lära sig om genom DataCamp. Det skulle ha varit en bra förberedelse inför kursen.
   Att läsa mer i kursboken – vid ett tillfälle kände jag att jag inte hängde med så mycket längre pga de matematiska bitarna och jag bör ha gått tillbaka till tidigare kapitel och repetera grundläggande koncepten igen.

   Att lägga mer tid i att faktiskt koda och leka med flera olika dataset – jag har inte kodat praktiskt tillräckligt, känner jag. Många syntax minns jag inte och behöver kolla upp hela tiden.