DV1621: Inlämningsuppgift 2

Maskininlärning med Python & Scikit-learn

Martin Boldt
Institutionen för Datavetenskap
Blekinge Tekniska Högskola

November 16, 2022

Introduktion

I denna inlämningsuppgift ska du använda maskininlärningsmiljön Scikit-learn[1] för Python. Med hjälp av Scikit-learn är det möjligt att lösa olika problem m.h.a. maskininlärning (ML). Scikit-learn är gratis och har öppen källkod (BSD-licens) och dessutom finns en bra community runt omkring, t.ex. exempelkod, diskussionsforum, turorials etc. Scikit-learn har därför blivit de facto standardmiljö i mycket maskininlärningsarbete, och är därför en logisk startpunkt för er när ni ska börja närma er området.
Denna inlämningsuppgift har som mål att ge er en första grundläggande förståelse för några av de viktigaste stegen som man behöver gå igenom för att praktiskt kunna använda maskininlärning för att lösa olika problem, t.ex. för klassifikation och klustring.
Ni kommer i den här uppgiften lösa ett par olika klassifikationsproblem med olika inlärningsalgoritmer. Datamängden ni kommer använa är Titanic-datamängden som ni fick se en liten del av i inlämningsuppgift 1.
Uppgiften delas upp i följande tre huvudsakliga delar: (i) uppföljning av K-nearest neighbors, (ii) Utforskande av datan och (iii) Klassifikation. Slutligen finns en sektion med extrauppgifter för er som vill uppnå högre betyg på denna uppgift (alltså betygen A eller B). Strävarna ni efter betygen C, D eller E behöver ni alltså inte göra extrauppgifterna.

Utförande och examinering

Deadline för uppgiften hittar ni på Canvas. Uppgifterna ska lösas parvis där en av er lämnar in en gemensam ZIP-fil som innehåller er Python-kod och en textrapport baserad på rapportmallen som finns på kurssidan. Inlämningsuppgiften bedöms enligt betygsskalan A-F. För att kunna få betygen A eller B behöver de frivilliga extrauppgifterna (som beskrivs i slutet av detta dokument) lösas.
Innan ni skickar in er inlämningsuppgift skall ni kontrollera att ni:

- använder den obligatoriska rapportmallen som finns på kurssidan,
- utförligt besvarat alla uppgifters deluppgifter,
- lämnar in en ZIP-fil som innehåller både er Pythonkod och er skriftliga rapport baserad på rapportmallen, och
- att ni skriver båda gruppmedlemmars för- och efternamn i er rapport och samtliga Python-filer.

[1] `https://scikit-learn.org/stable/`

Installation

En förutsättning för att kunna använda Scikit-learn är att du har tillgång till en fungerande installation av Python 3. När du säkerställt att du har detta så är nästa steg att installera Scikit-learn. Använd det pakethantreingssytem du är van vid att använda, t.ex. `conda` eller `pip3`. För att exempelvis installera Scikit-learn m.h.a. `pip3` så skriver du följande kommando i DOS-prompten/Powershell för Windows, alternativt i Terminalen i Unix eller Mac OS X:

```
pip3 install scikit-learn
```

Om pakethanteringssystemet klagar på att något paket krävs av Scikit-learn, men saknas, så installera det först på samma sätt som ovan. Försök därefter installera Scikit-learn igen.
När du har installerat Scikit-learn så är nästa steg att ladda ner Titanic-datamängden från kurssidan. Ni ska ladda ner följande tre filer som finns publicerade under Inlämningsuppgift 2 på Canvas.

- `1-titanic-small.csv`: datamängden ni använde i inlämningsuppgift 1.
- `2-titanic2attr.csv`: samtliga instanser, men endast de två attribut/kolumner som ni använde i inlämningsuppgift 1.
- `3-titanic.csv`: samtliga instanser och attribut.

För en introduktion till Scikit-learn så behöver ni gå igenom följande tutorial som går igenom inläsning av datasets samt klassifikation av datan[2]. På denna länk[3] finns även ett par praktiska exempel som visar inläsning av data från CSV-fil, pre-processing samt klassificering och en enkel utvärdering. När ni känner att ni har en övergripande förståelse för hur Scikit-learn fungerar kan ni fortsätta med att lösa uppgifterna nedan.

Del 1: K-nearest neighbors

Inledningsvis ska ni börja med att utföra klassifikation med KNN-algoritmen på den lilla delmängd av titanic-datasetet som ni använde i inlämningsuppgift 1. Mer specifikt är det de 10 första raderna i Tabell 2 från inlämningsuppgift 1. Sedan ska ni använda utföra samma klassifikation på en större del av datasetet samt det fullständiga datasetet för att utvärdera skillnaden i prestanda.

Uppgift 1.1

Första deluppgiften går ut på att m.h.a. Scikit-learn beräkna hur bra [4] KNN kan klassificera baserat på den lilla datamängd ni använda i inlämningsuppgift 1. Börja med att läsa in den datamängden (som finns i filen: `1-titanic-small.csv`) till en Pandas DataFrame m.h.a. funktionen `read_csv()`. För dokumentation kring hur olika funktioner fungerar så använd relevant dokumentation online. Exempelvis genom att söka med Google på t.ex. `"pandas read_csv"` för att hitta manual och kodexempel.
När datan är inläst så är nästa steg att initiera KNN-inlärningsalgoritmen. För att göra detta använder ni funktionen `KNeighborsClassifier()` som finns i biblioteket `sklearn.neighbors`. Sätt värdet på `K` till 3 genom att ändra KNN:s hypterparameter med namn `n_neighbors` såhär:

```
knn = KNeighborsClassifier(n_neighbors=3)
```

[2] `https://scikit-learn.org/stable/tutorial/basic/tutorial.html`
[3] `https://www.activestate.com/resources/quick-reads/how-to-classify-data-in-python/`
[4] Måttet som används kommer vara accuracy (träffsäkerhet på svenska).

Koden ovan returnerar ett objekt som innehåller en KNN-modell som är konfigurerad att analysera de tre närmsta grannarna vid klassifikation av instanser. Nästa steg i processen är att anpassa den modellen till just specifikt det data vi läst in från filen. Detta gör man med funktionen `knn.fit()`. Den funktionen behöver följande två inparametrar: `X` och `y`. `X` ska vara en 2-dimensionell matris med all data (raderna = instanser, och kolumner = attribut). Dock ska inte klassvariabeln (det attribut som heter `Survived` finnas med i `X`). Värdena för klassvariabeln ska istället sparas ner i `y` som en array. Exemplet nedan visar ett `X` som består av tre instanser (den första med åldern 49 samt 0 syskon). Arrayen `y` specificerar att den första och tredje instansen överlevde (`Survided=1`) medan den andra omkom (`Survided=0`).

```
X = [ [49 , 0], [35 , 2], [12 , 1] ]
y = [1, 0, 1]
```

Nästa steg är alltså att omvandla den inlästa DataFramen till `X` och `y`. Ett tips är att använda Pandas `iloc[]`-funktion för att göra detta. När ni har skapat `X` och `y` så använder ni `fit()`-funktionen för att anpassa er KNN-modell till datan.

Nu har ni en tränad KNN-modell som ni kan använda för att klassificera huruvida instanser sannolikt överlevde eller ej. För att göra det använder ni funktionen `predict()` som tar samma typ av instansdata som finns i `X` som input. Nedan visas ett kodexempel som klassificerar de tre exempel-instanserna från exemplet ovan:

```
# Skapa de tre instanserna
new_instances = [ [49 , 0], [35 , 2], [12 , 1] ]

# klassificera dem m.h.a. modellens predict()-funktion
y_predicted = knn.predict( new_instances )
```

Resultatet som lagras i arrayen `y_predicted` är följande `[1, 0, 1]`. Resultatet anger klassvariablen för var och en av de tre helt nya instanserna. Den andra instansen klassificerades alltså som omkommen, medan övriga två som överlevande. Använd nu funktionen `predict()` för att se hur er KNN-modell klassificerar samtliga instanser i den inlästa datamängden. Lagra resultatet i `y_predicted` för samtliga 10 instanser.
Sista steget i denna deluppgift är att beräkna modellens accuracy (träffsäkerhet) baserat på klassvariabelns korrekta värdena (alltså "facit" som finns i `y`) respektive modellens klassificerade värden i `y_predicted`. Ett tips är att använda funktionen `accuracy_score()` som finns i biblioteket `sklearn.metrics`, vilken tar som input just `y`) och `y_predicted`. Skriv ner er uppmätta accuracy i laborationsrapporten under sektion 1.1.

Uppgift 1.2

Läs in datamängden `2-titanic-2attr.csv` som fortfarande bara innehåller två attribut, men samtliga instanser som finns i Titanic-datamängden. Eftersom vissa av attributvärdena är tomma (saknar mätvärde), t.ex. om det saknas ålder för en viss instans. Därför behöver dessa instanser plockas bort först då KNN-algoritmen annars inte kan räkna ut ett avstånd till dessa instanser för att se vilka andra instanser som är närmast. För att ta bort sådana instanser så använd `dropna()`-funktionen. I exemplet nedan tas alla rader som saknar minst ett värde (såkallade `NaN`: not a number) bort från DataFramen. Därefter sparas den nya DataFramen med samma namn.

```
titanic2attr = titanic2attr.dropna()
```

Efter att ni plockat bort saknade värden så använder ni samma procedur som i föregående uppgift för att beräkna accuracy baserat på den nya datan. Skriv ner accuracyn i er laborationsrapport.

Uppgift 1.3

Läs slutligen in den fullständiga datamängden `3-titanic.csv` och utför samma procedur igen, bibehåll $K = 3$. Men, ett problem är att fyra av attributen (nämligen `Fare`, `Embarked`, `Sex` och `Ticket`) behöver förprocesseras få att KNN-algoritmen ska kunna tolka dem. Dessa innehåller bokstavsvärden vilka behöver representeras numeriskt istället. För att lösa detta använder man Scikit-learns `LabelEncoder()`-funktion (som finns i biblioteket `sklearn.preprocessing`). Med hjälp av LabelEncodern kan ni därefter kan anropa funktionen `transform()` för att transformera textvärden till numeriska värden istället. Mer information om hur man gör detta finns i Scikit-learns dokumentation.
Efter att ni transformerat datan med LabelEncodern så använder ni precis som tidigare `fit()` och `predict()` för att därefter beräkna modellens accuracy. Skriv ner vilken accuracy som uppmäts i laborationsrapporten.

Uppgift 1.4

Jämför de olika värdena av accuracy som ni uppmätt i Uppgift 1.1 - 1.3. Vad beror skillnaden i accuracy på och vad drar ni för slutsats av resultatet? Ser ni några problem med hur ni har tränat och utvärderat era KNN-modeller ovan? Beskriv i så fall kortfattat vad ni anser är problem.

Del 2 - Utforskande av datan

För att få ut de bästa resultaten gäller det att noga analysera och förstå den data man har att arbeta med. Försäkra er om att ni använder den stora datamängden (filen `titanic.csv`). Bekanta er med vilken data som finns i DataFramen efter att ni läst in CSV-filen. Exempel på användbara funktioner för att få insikt i datan är: `shape`, `dtypes` och `Describe()`.

Uppgifter

Efter att ni har bekantat er lite närmare med datan ska ni besvara nedanstående frågor:

1. Hur stort är datasetet, dvs. hur många instanser innehåller det, och hur många attribut beskriver instanserna?

2. Undersök de olika attributen och vilken typ de utgör. Vilka olika typer består datasetet av? Hur många attribut utgör respektive typ? Förklara skillnaden mellan typerna.

3. Undersök attributet *Sex* och skriv ut hur många av respektive kön som datamängden innehåller. Skriv även ut hur många procent av respektive kön som omkom (`Survived==0`).

4. Undersök klassattributet *Survived* och avgör vilken klass-ratio datasetet har. Är datasetet balanserat (ungefär lika många instanser av varje klass) eller obalanserat?

Del 3 - Klassifikation

I denna del ska ni utvärdera hur bra Scikit-learns inlärningsalgoritmer `DecisionTree()` respektive `RandomForest()` presterar på den stora Titanic datamängden. Använd de förinställda värdena för algoritmernas hyperparametrar. När ni utvärderar algoritmerna ska ni dessutom dela upp datan i två delar så ni slumpvis använder 70 % att träna modellerna på, och resterande 30 % av datan för att utvärdera dem genom att beräkna dess accuracy. Använd funktionen `train_test_split()` för att splitta upp datan i träning- och testdata.

Uppgifter

1. Vilken accuracy uppnår DecisionTree-modellen?
2. Vilken accuracy uppnår RandomForest-modellen?
3. Vad tror ni skillnaden mellan dem beror på?

Extrauppgifter för högre betyg

Följande uppgifter är inte obligatoriska men är tillgängliga för er som vill utmana er ytterligare för att ha möjlighet att uppnå ett högre betyg (betyg A och B).

4. Är accuracy alltid ett bra mätvärde? Finns det något annat mätvärde som presenteras som kan vara mer lämpligt än accuracy baserat på hur klass-ration ser ut (alltså fördelningen mellan antalet instanser i respektive klass)? Beskriv vilket mätvärde och varför det är mer passande.
5. Beräkna detta nya mätvärde för dina DecisionTree och RandomForest-modeller och undersök hur de förhåller sig till accuracy-måtten.
6. Försök uppnå så hög prestanda som möjligt med det mätvärde ni hittade i extrauppgiften ovan (alltså inte accuracy). Ni har fria händer att välja vilken algoritm ni vill (så länge de ingår i Scikit-learns standard bibliotek) samt att utföra vilken preprocessing ni vill (filter, borttagning av attribut, etc). Ni ska också prova olika hyperparametrar för er algoritm. Beskriv och motivera de val ni gjort. Vad funkade exempelvis bra och vad funkade mindre bra? Reflektera över varför. Tips: sök på internet efter hyperparametrar till er valda algoritm för att få förståelse om dem och för att få tips om vilka som kan vara bra att justera.

Lycka till!