Relazione progetto Data Mining
Shelter Animal Outcomes

GIACOMO MANZOLI
1130822
Università degli Studi di Padova
15 giugno 2016

Indice

1 Introduzione

1.1 Descrizione del dataset

Il dataset contiene 26729 osservazioni relative agli animali che hanno lasciato il rifugio per animali della città di Austin nel periodo che va dall'Ottobre 2013 a Marzo 2016. L'obiettivo è quello di utilizzare i dati del dataset per prevedere quale sarà il destino dei nuovi animali che verranno accolti nel centro.

I dati sono forniti da Kaggle per la competizione "*Shelter Animal Outcomes*"[1] e, trattandosi di una sfida, viene fornito anche un secondo dataset di 11456 osservazioni, per le quali non è nota la variabile risposta, che deve essere utilizzato per fornire al sito le proprie previsioni, al fine di stilare una classifica.

1.2 Descrizione delle variabili

Il dataset è composto da 10 variabili che descrivono lo stato dell'animale quando ha lasciato il rifugio. Più nel dettaglio:

- `AnimalID`: codice univoco che viene affidato all'animale quando è entrato nel rifugio. Nel dataset principale viene fornito sotto forma di stringa, mentre nel dataset secondario viene fornito come numero intero.
- `Name`: nome dell'animale. Nel dataset principale ci sono 7691 animali senza un nome.
- `DateTIme`: data e ora in cui l'animale ha lasciato il rifugio. È espressa nel formato `aaaa-mm-gg hh:mm:ss`.
- `Outcome`: variabile risposta, ha 5 possibili valori:
  - *Adoption*: 10769 osservazioni.
  - *Died*: 197 osservazioni.
  - *Euthanasia*: 1555 osservazioni.
  - *Return to owner*: 4786 osservazioni.
  - *Transfer*: 9422 osservazioni.
- `OutcomeSubtype`: variabile che descrive perché l'animale ha fatto quella particolare fine. Ci sono 17 possibili valori per questa variabile e per 13612 non è disponibile. Questa variabile non è presente sul dataset secondario.
- `AnimalType`: tipo dell'animale, può essere un cane o un gatto.
- `SexuponOutcome`: sesso dell'animale, comprende anche l'informazione se l'animale è stato castrato o meno. In tutto ci sono 6 possibili valori per questa variabile.
- `AgeuponOutcome`: età dell'animale quando ha lasciato il rifugio, viene espressa utilizzando una stringa che descrive l'età, ad esempio: *2 years, 1 week, ecc.*
- `Breed`: razza dell'animale. Comprende anche l'informazione se l'animale è un incrocio di più razze e in qualche caso specifica anche la seconda razza. In tutto ci sono 1380 possibili valori.
- `Color`: colore del pelo dell'animale. Comprende anche le informazioni relative al pelo e ad un eventuale colore secondario. In tutto ci sono 336 possibili valori.

La struttura del dataset una volta caricato in R è la seguente:

[1] `https://www.kaggle.com/c/shelter-animal-outcomes`

```
'data.frame': 26729 obs. of  10 variables:
$ AnimalID      : Factor w/ 26729 levels "A006100","A047759",..
$ Name          : Factor w/ 6375 levels "","Joanie","Mario",..
$ DateTime      : Factor w/ 22918 levels "2013-10-01 09:31:00",..
$ OutcomeType   : Factor w/ 5 levels "Adoption","Died",..
$ OutcomeSubtype: Factor w/ 17 levels "","Aggressive",..
$ AnimalType    : Factor w/ 2 levels "Cat","Dog"
$ SexuponOutcome: Factor w/ 6 levels "","Intact Female",..
$ AgeuponOutcome: Factor w/ 45 levels "","0 years","1 day",..
$ Breed         : Factor w/ 1380 levels "Abyssinian Mix",..
$ Color         : Factor w/ 366 levels "Agouti","Agouti/Brown Tabby",..
```

Inoltre, andando a tracciare il grafico con le proporzioni delle varie classi in base al tipo di animale (Figura 1) è possibile osservare che per i cani è più probabile che siano recuperati dai propri padroni rispetto ai gatti, mentre per i gatti è più probabile che vengano trasferiti.

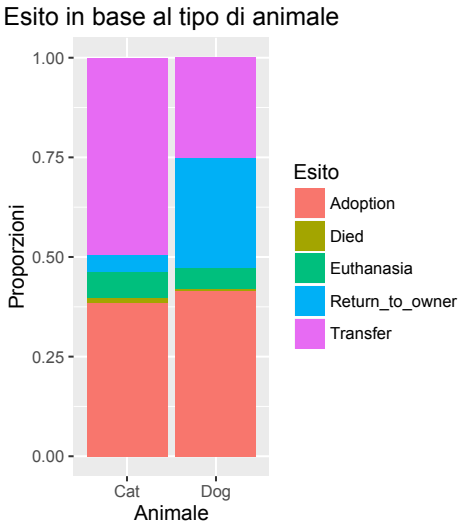

Figura 1: Esito in base al tipo di animale

1.3 Software utilizzato

Nella realizzazione del progetto è stato utilizzato l'ambiente R in versione 3.2.4, esteso con alcune librerie:

- `ggplot2`: per la rappresentazione dei grafici.
- `dplyr`: per la manipolazione dei dati.
- `nnet`: reti neurali e regressione logistica.
- `GAM`: GAM.
- `earth`: MARS.
- `randomForest`: random forest.
- `tree`: alberi di classificazione.
- `ada`: Boosting.

Tutte le librerie sono disponibili su CRAN mentre il codice del progetto è disponibile su GitHub all'indirizzo `https://github.com/GiacomoManzoli/AnimalShelter`.

2 Elaborazione delle variabili

Il dataset si presenta con poche variabili che riassumono molte informazioni o che hanno un numero molto elevato di livelli. È quindi necessario andare ad estrapolare le informazioni da queste variabili, creandone di nuove e di più semplici.

2.1 Il nome dell'animale

La variabile `Name` ha più di 6000 possibili valori distinti e, ragionando a livello di previsione, sembra poco probabile che il nome dell'animale influisca sul suo destino. Tuttavia ci sono 7691 animali che non hanno un nome, questo probabilmente implica che si tratta di animali randagi che sono stati portati al rifugio[2] e quindi può essere meno probabile che il padrone li venga a recuperare. Analogamente se è stato trovato un'animale smarrito e con una targhetta con il nome al collo, è più probabile che il suo padrone vada a recuperarlo.

Il nome può quindi essere riassunto da una nuova variabile booleana `HasName` che specifica se l'animale ha un nome o meno.

Tracciando il grafico (Figura 2) per visualizzare la ripartizione delle varie classi per la variabile risposta in base al valore di `HasName` si ha che le osservazioni fatte sembrano essere confermate dai dati: indipendentemente dal tipo di animale, se questo ha un nome è più probabile che venga recuperato dal suo padrone. Inoltre, si può notare anche che per i gatti, la probabilità di essere adottati è molto più alta se hanno un nome.

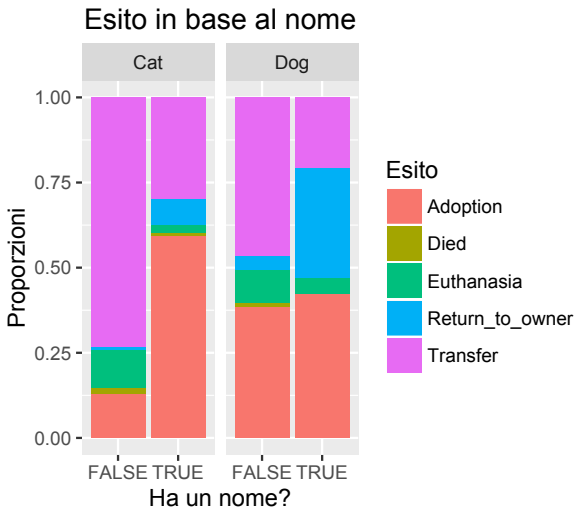

Figura 2: Esito in base al nome

2.2 La data di uscita

Difficilmente la data in cui l'animale ha lasciato la struttura può tornare utile per effettuare previsioni future. Si possono però estrarre altre informazioni come la fascia oraria e il giorno della settimana in cui l'animale ha lasciato la struttura.

Queste due informazioni possono tornare utili perché, ad esempio la gente nei week.end ha più tempo libero e quindi è più probabile che vada ad adottare un animale, oppure che per motivi logistici i trasferimenti possono essere fatti solamente la mattina.

Per estrapolare ciò vengono definite due nuove variabili `DayOfWeek` che assume come valore il giorno della settimana e `TimeOfDay` che può assumere come valori:

[2]Non vengono fornite informazioni a riguardo

- *Mattina*: se l'ora è compresa tra le 6 e le 12.
- *Pomeriggio*: se l'ora è compresa tra le 12 e le 17.
- *Sera*: se l'ora è compresa tra le 17 e le 20.
- *Notte*: per le restanti ore.

Come si può notare dal grafico (Figura 3) c'è un picco sulle adozioni nella fascia serale. Sempre dal grafico si può notare che la notte sono più frequenti i trasferimenti, anche se la maggior parte di questi si viene svolta durante le ore diurne.

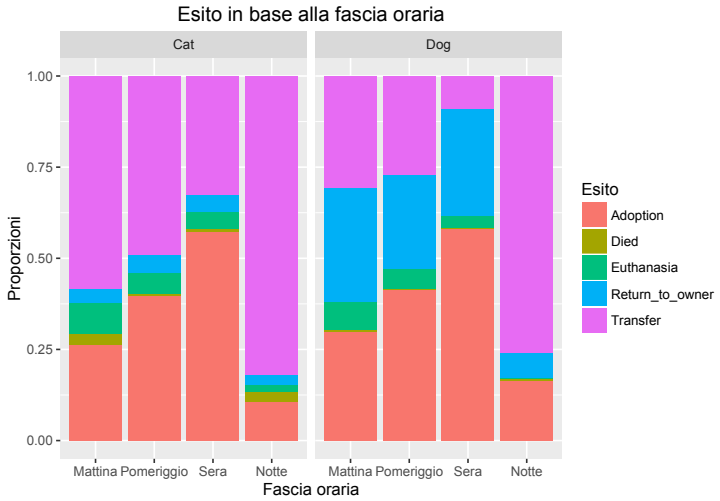

Figura 3: Esito in base alla fascia oraria

Per quanto riguarda il giorno della settimana, dal grafico Figura 4, si può notare come le adozioni siano leggermente più probabili nel week-end.

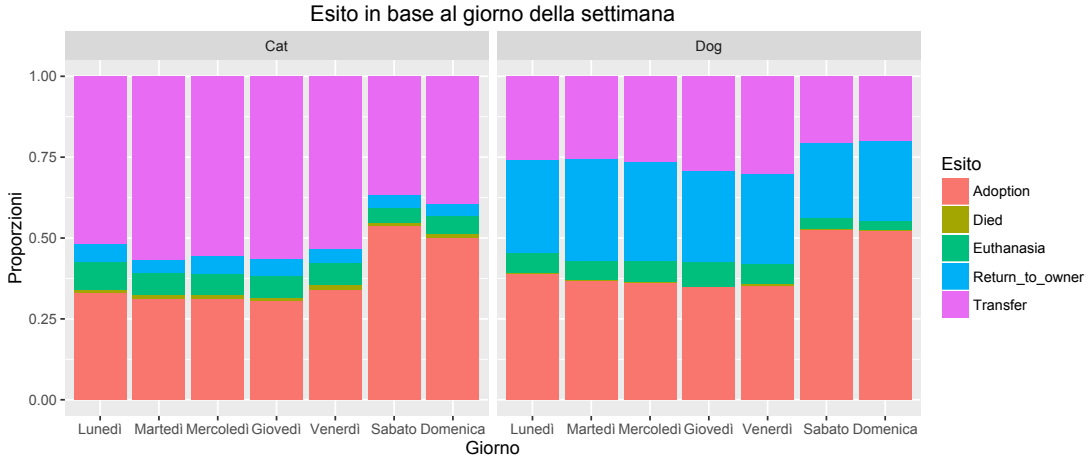

Figura 4: Esito in base al giorno della settimana

2.3 Sesso e stato dell'animale

La variabile `SexuponOutcome` prevede 6 possibili livelli che racchiudo l'informazione relativa al sesso e al fatto se l'animale è stato castrato o meno. C'è poi un livello *Unknown* per gli animali per i quali non si hanno informazioni e in più ci sono dei valori non disponibili.

Questa variabile è stata quindi scomposta in `Gender` che specifica il sesso dell'animale e `Status`, che specifica se l'animale è stato castrato o meno. Entrambe le variabili hanno un terzo possibile valore *Unknown* che rappresenta i dati non disponibili.

Dal grafico riportato in Fig 5 si può notare che indipendentemente dal sesso e dal tipo di animale, se l'animale è stato castrato è più probabile che venga adottato. Se invece non è stato castrato oppure non ci sono informazioni a riguardo, è più probabile che venga trasferito.

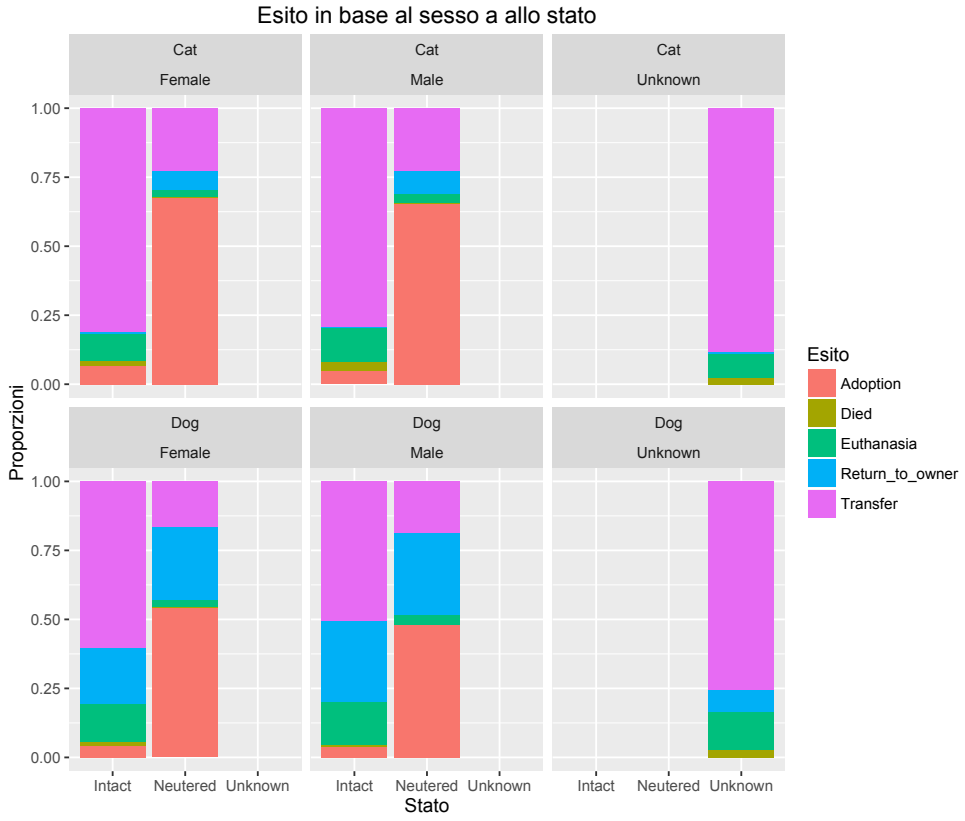

Figura 5: Esito in base al sesso e allo stato dell'animale

2.4 L'età dell'animale

L'età dell'animale è memorizzata nella variabile `AgeuponOutcome` in un formato molto confusionale in quanto viene espressa come *2 anni*, *3 mesi*, ecc. Inoltre ci sono dei casi in cui alcuni valori non hanno la *s* del plurale, ovvero tra i possibili valori della variabile ci sono ad esempio *1 week* e *1 weeks*, che quindi vengono considerati come valori distinti quando in realtà non lo sono.

La prima modifica è quindi quella di normalizzare i valori, esprimendoli con un numero intero che approssima l'età dell'animale espressa in giorni. Così facendo risulta più semplice classificare gli animali per fascia d'età. Infatti, si può assumere che un cucciolo è più probabile che venga adottato rispetto ad un animale più anziano, mentre gli animali troppo piccoli non possono essere adottati per legge.

Conviene quindi creare una nuova variabile `AgeCategory` con 5 possibili livelli:

- *Neonato*: da 0 a 29 giorni.
- *Cucciolo*: da 30 a 365 giorni.
- *Adulto*: da 366 a 3650 giorni (10 anni).
- *Anziano*: più di dieci anni.

- *Sconosciuta*.

Nella normalizzazione dei valori si è scelto di mantenere le 18 osservazioni con i valori mancati per l'età, marcandoli come sconosciuti, questo perché nel secondo dataset sono presenti delle osservazioni per le quali l'eta non è nota. Inoltre, si può ipotizzare che questi animali siano randagi e quindi che per questo motivo la loro eta non è nota. Questa ipotesi deriva dal fatto che tra i possibili valori della variabile `OutcomeSubtype` c'è il valore *SCRP* che indica un trasferimento relativo al programma di recupero dei gatti randagi[3] e quasi tutte le osservazioni con l'età mancante hanno proprio quel valore come `OutcomeSubtype`.

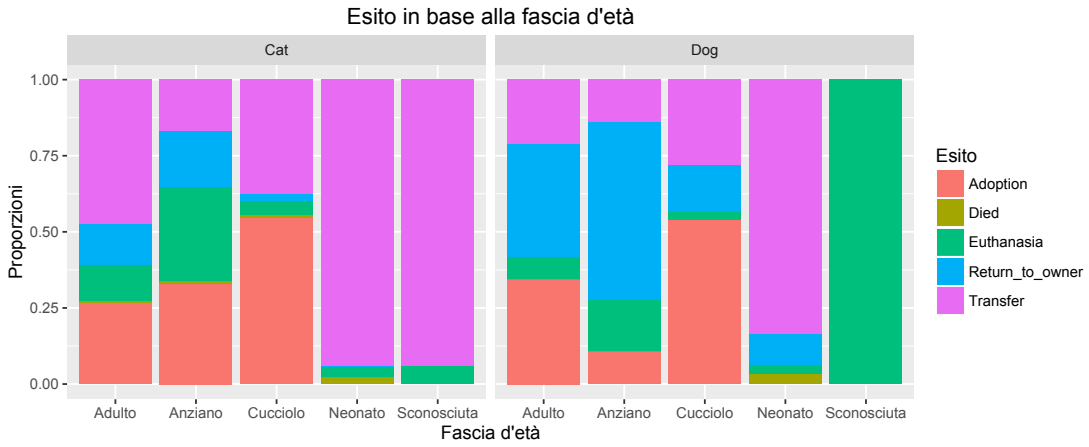

Figura 6: Esito in base alla fascia d'età dell'animale

Come si può notare dal grafico Figura 6, nessuno degli animali neonati viene adottato ed è più probabile che vengano trasferiti. Per quanto riguarda la probabilità di adozione, questa è maggiore per i cuccioli e più bassa per i cani anziani. I gatti anziani hanno una maggiore probabilità di adozione rispetto ai cani anziani e questo può essere dovuto al fatto che la speranza di vita di un gatto è maggiore rispetto a quella di un cane.

2.5 Razza

Le informazioni relative alla razza dell'animale sono racchiuse nella variabile `Breed`, la quale ha 1340 possibili valori e specifica anche se l'animale è un incrocio o meno.

Osservando alcuni dei possibili valori, si può notare che se l'animale non è di razza, il valore della variabile comprende o due razze oppure la razza principale, seguita da *Mix*. Si è scelto quindi di scomporre la variabile `Breed` nelle variabili `PrimaryBreed` (220 livelli), `SecondaryBreed` (144 livelli) e `IsMix` (booleana).

Ci sarebbero ulteriori informazioni che possono essere estratte da questa variabile, come la stazza dell'animale, la quale a sua volta va ad influire sull'aspettativa di vita e quindi sulla corretta classificazione della fascia d'età e sul carattere dell'animale. Tuttavia per estrarre queste informazioni in modo corretto è necessaria un'elevata conoscenza del dominio, ci si è quindi limitati alla scomposizione della variabile `Breed`.

Come si può notare dal grafico in Figura 7, un cane di razza ha più probabilità di essere adottato rispetto ad un cane non di razza, mentre per i gatti sembra che avvenga il contrario. Non sono stati inseriti i grafici per le variabili `PrimaryBreed` e `SecondaryBreed` perché il numero elevato di possibili valori li rende incomprensibili.

[3] `http://www.maddiesfund.org/austin-animal-services-stray-cat-return-program.htm`

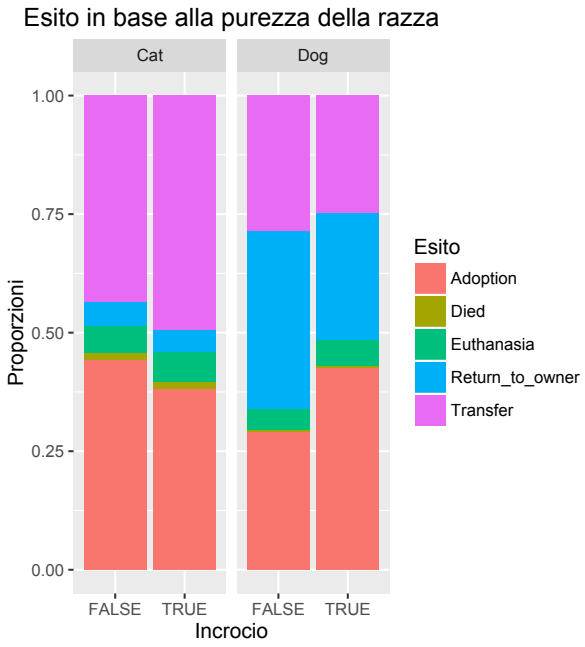

Figura 7: Esito in base alla fascia d'età dell'animale

2.6 Colore

Come per la razza, anche per il colore le informazioni sono racchiuse nell'unica variabile `Color` e, sempre come per la razza, queste informazioni sono state suddivise nelle variabili:

- `PrimaryColor`: colore principale, 29 livelli.
- `SecondaryColor`: colore secondario, 24 livelli.
- `Pattern`: pattern del pelo, 10 livelli.
- `HasComplexColor`: valore booleano che specifica se il pelo dell'animale ha più colori o un certo pattern particolare.

Sarebbe poi necessario andare a normalizzare i valori dei colori, dato che ci sono più livelli che indicano lo stesso colore, come *Orange* e *Apricot* (albicocca), ma da una prima analisi grafica (Figura 8) sembra che, fissato il tipo di animale, le informazioni sul colore non influiscano sull'esito e in quei pochi casi che questo succede può essere dovuto al fatto che si hanno troppe poche osservazioni con quel determinato colore.

2.7 Riassunto delle trasformazioni

Dopo aver applicato le trasformazioni precedentemente descritte ed aver eliminato le variabili `AnimalID` e `OutcomeSubtype`, il dataset ha assunto la seguente struttura:

```
'data.frame': 26711 obs. of  15 variables:
$ OutcomeType    : Factor w/ 5 levels "Adoption","Died",..
$ AnimalType     : Factor w/ 2 levels "Cat","Dog"
$ AgeCategory    : Factor w/ 4 levels "Adulto","Anziano",..
$ DayOfWeek      : Factor w/ 7 levels "Lunedì","Martedì",..
$ TimeOfDay      : Factor w/ 4 levels "Mattina","Pomeriggio",..
$ Gender         : Factor w/ 3 levels "Female","Male",..
$ Status         : Factor w/ 3 levels "Intact","Neutered",..
```

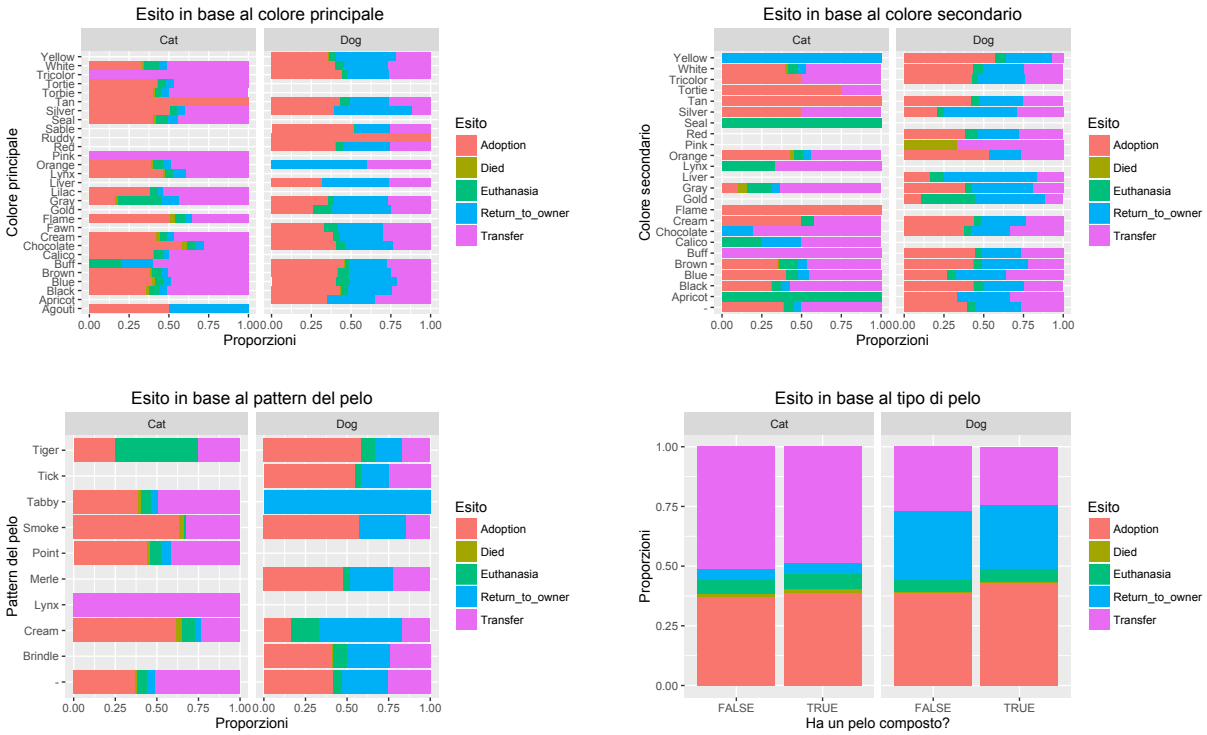

Figura 8: I grafici della prima riga rappresentano rappresentano l'esito in base al colore principale e secondario. I grafici della seconda riga rappresentano l'esito in base al pattern e al fatto se il pelo dell'animale è composto o meno. Salvo alcuni casi dovuti al fatto che sono presenti poche osservazioni con quel determinato valore per una delle variabili, non si nota una correlazione tra il pelo e l'esito dell'animale una volta stabilito se si tratta di un cane o un gatto.

```
$ PrimaryColor   : Factor w/ 29 levels "Agouti","Apricot",..
$ SecondaryColor : Factor w/ 24 levels "-","Apricot",..
$ Pattern        : Factor w/ 10 levels "-","Brindle",..
$ HasComplexColor: Factor w/ 2 levels "FALSE","TRUE"
$ PrimaryBreed   : Factor w/ 220 levels "Abyssinian","Affenpinscher",..
$ SecondaryBreed : Factor w/ 144 levels "-","Affenpinscher",..
$ IsMix          : Factor w/ 2 levels "FALSE","TRUE"
$ HasName        : Factor w/ 2 levels "FALSE","TRUE"
```

3 Modelli

Una volta trasformati i dati, sono stati provati vari modelli per vedere quale riesce ad effettuare le previsioni migliori. In particolare sono stati utilizzati:

- Regressione logistica multiclasse
- MARS
- GAM
- Alberi di classificazione
- Random Forest
- Reti Neurali
- Boosting

Per calcolare e confrontare i modelli, le 26000 osservazioni sono state suddivise in due insiemi, un insieme di validazione contenente il 20% delle osservazioni e che viene utilizzato per confrontare gli errori commessi dai vari modelli e un altro set di dati composto dal restante 80% delle osservazioni.

Le osservazioni di quest'ultimo set sono poi state suddivise in altri due sottoinsiemi, il *test set* contenete il 20% e il *train set* contenente le restanti osservazioni.

Con questa suddivisione è possibile utilizzare il *train set* per calcolare il modello, provando più valori per gli eventuali iper-parametri, il *test set* per confrontare quale configurazione di iper-parametri funziona meglio ed infine combinare i due set per calcolare la versione del modello da confrontare con gli altri modelli sui dati del *validation set*.

Infine, una volta trovata la configurazione ottimale di ogni modello, questo viene ricalcolato anche sulla totalità delle osservazioni per poi essere utilizzato per effettuare le previsioni sul dataset secondario in modo da poterle caricare su Kaggle, ottenendo così un'ulteriore metrica per la valutazione della bontà del modello.

Non è precisato su che cosa si basa il punteggio attribuito da Kaggle, tuttavia minore è il valore attribuito, migliore è il modello e per entrare nella top 100 dei punteggi migliori è necessario scendere sotto lo 0.7236.

Per quanto riguarda le librerie contenenti le implementazioni dei vari modelli, si è cercato di utilizzare quelle viste a lezione, tuttavia non tutte queste librerie sono in grado di gestire la classificazione multiclasse e quindi è stato necessario cercare delle versioni alternative. Per i modelli GAM e Boosting, non sono state trovate librerie alternative, sono stati quindi utilizzati più modelli binomiali, combinati secondo la strategia *one-vs-all*.

3.1 Regressione logistica multiclasse

Il primo modello utilizzato è stato quello che effettua la regressione logistica multiclasse. Tra le varie implementazioni della regressione logistica disponibili per R si è scelto di utilizzare `multinom` presente nel pacchetto `nnet`, la quale simula una regressione logistica utilizzando una rete neurale.

È stata scelta questa particolare implementazione perché la funzione `glm` utilizzata in laboratorio funziona solo per la regressione binomiale e la funzione `mlogit` dell'omonimo pacchetto si è rilevata eccessivamente complessa.

Il modello calcolato da `multinom` prevede, tra i vari iper-parametri, il valore di `decay` e confrontando l'effetto dei vari valori sul *test set*, si è trovato come valore migliore 0.001, il quale è stato utilizzato per calcolare il modello di regressione sulla versione combinata del *test* e *train set*, producendo un errore sul test di classificazione pari a 0.3365762, ovvero circa del 33.66%.

Gli errori di classificazione sono riportati nella Tabella 1, dalla quale si può notare che la maggior parte degli errori commessi riguardano i trasferimenti che vengono classificati erroneamente come adozione. Un'altra cosa che si può notare è che nessuno degli animali viene classificato come deceduto per morte naturale e questo può essere causato dal fatto che nel dataset ci sono poche osservazioni che descrivono animali morti per cause naturali.

Per quanto riguardo il punteggio ottenuto su Kaggle, il modello calcolato su tutto il dataset principale ha ottenuto un punteggio di 0.89358.

3.1.1 Regressione utilizzando `glm`

Per effettuare la regressione logistica utilizzando `glm` è stato necessario calcolare 5 modelli, uno per ogni possibile valore della variabile risposta, e combinare tra loro le probabilità ottenute in modo da ottenere la classe più probabile per una determinata osservazione.

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1827	1	38	364	462
	Died	0	0	0	0	0
	Euthanasia	3	1	40	17	21
	Return to owner	242	4	71	446	142
	Transfer	120	36	173	104	1233
	Errore classe (%)	16.65	100	87.57	52.09	33.63

Tabella 1: Errori di classificazione con il modello di regressione logistica.

Con questo modello si è ottenuto un errore del 39.86% e un punteggio su Kaggle di 0.87527, anche se durante il calcolo del modello completo vengono sollevati dei warning rivelativi all'utilizzo di matrici con un rango troppo basso.

La Tabella 2 riporta le matrici di confusione per i 5 modelli binomiali, mentre la Tabella 3 riporta il resoconto degli errori di classificazione. Da entrambe le tabelle è possibile notare come questo modello classifichi erroneamente molte osservazioni con la classe *Died*, mentre la versione del pacchetto `multinom` non classifica nessuna osservazione con questa classe.

		Valori predetti	
		No	Yes
Valori osservati	No	2512	641
	Yes	522	1670

(a) OutAdoption, errore: 0.217

		Valori predetti	
		No	Yes
Valori osservati	No	4766	537
	Yes	40	2

(b) OutDied, errore: 0.108

		Valori predetti	
		No	Yes
Valori osservati	No	5008	15
	Yes	298	24

(c) OutEuthanasia, errore: 0.058

		Valori predetti	
		No	Yes
Valori osservati	No	4279	135
	Yes	727	204

(d) OutReturnToOwner, errore: 0.161

		Valori predetti	
		No	Yes
Valori osservati	No	3175	312
	Yes	738	1120

(e) OutTransfer, errore: 0.196

Tabella 2: Matrici di confusione per i 5 classificatori con GLM

3.2 MARS

Come modello MARS è stato utilizzato quello disponibile nel pacchetto `earth` con la configurazione di default.

Così facendo si è ottenuto un errore del 36.40% e un punteggio su Kaggle di 0.9001.

Anche in questo caso, come si può notare dalla Tabella 4, il modello non ha classificato nessuna delle osservazioni come *Died* o *Euthanasia*.

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1681	2	43	344	455
	Died	238	2	29	124	120
	Euthanasia	0	1	17	4	11
	Return to owner	184	2	59	362	120
	Transfer	89	35	174	97	1152
	Errore classe (%)	23.31	95.23	94.72	61.11	37.66

Tabella 3: Errori di classificazione con GLM.

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1994	2	61	599	566
	Died	0	0	0	0	2
	Euthanasia	0	0	0	0	0
	Return to owner	99	2	48	187	74
	Transfer	99	38	213	145	1218
	Errore classe (%)	9.03	100	100	79.91	34.44

Tabella 4: Errori di classificazione con MARS.

3.3 GAM

Il modello GAM è stato calcolato utilizzando la funzione `gam` dell'omonimo pacchetto e selezionando tutte le variabili a disposizione.

Dal momento che GAM utilizza come predittore un modello lineare generalizzato e che non è stato possibile utilizzare le spline di lisciamento perché tutte le variabili sono qualitative, i risultati ottenuti con questo modello sono molto simili a quelli ottenuti con il modello lineare generalizzato riportato in §3.1.1.

Sempre come per il modello della regressione logistica, non è stato possibile calcolare un unico modello, ma è stato necessario definire 5 modelli, uno per ogni possibile valore della variabile risposta, per poi classificare l'osservazione utilizzando la classe più probabile.

Così facendo si è ottenuto un errore di classificazione del 34.03% sul test di validazione e un punteggio di 0.87405 su Kaggle.

Gli errori di classificazione vengono riportati nelle tabelle 5 e 6. Dalla prima tabella è possibile osservare come il modello `OutDied` non classifichi nessuna osservazione come positiva, ottenendo comunque un errore molto basso a causa del numero limitato di osservazioni che compaiono nel training set che hanno quella determinata classe.

3.4 Alberi di classificazione

Per calcolare l'albero di classificazione è stato utilizzato il modello disponibile nel pacchetto `tree`.

L'albero è stato costruito utilizzando le osservazioni presenti nel *train set*, espandendolo fino ad ottenere una devianza interna alle foglie minore di 0.002 (`mindev=0.002`). Dopodiché sono stati usati i dati del *test set* per potare l'albero in modo da limitare l'overfitting, ottenendo come albero migliore quello con 21 foglie.

L'albero così prodotto ha ottenuto un errore di classificazione del 35.43%, ovvero leggermente peggiore rispetto alla regressione logistica,

		Valori predetti	
		No	Yes
Valori osservati	No	2512	641
	Yes	522	1670

(a) OutAdoption, errore: 0.217

		Valori predetti	
		No	Yes
Valori osservati	No	5303	0
	Yes	42	0

(b) OutDied, errore: 0.007

		Valori predetti	
		No	Yes
Valori osservati	No	5008	15
	Yes	298	24

(c) OutEuthanasia, errore: 0.058

		Valori predetti	
		No	Yes
Valori osservati	No	4279	135
	Yes	727	204

(d) OutReturnToOwner, errore: 0.161

		Valori predetti	
		No	Yes
Valori osservati	No	3175	312
	Yes	738	1120

(e) OutTransfer, errore: 0.196

Tabella 5: Matrici di confusione per i 5 classificatori con GAM

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1878	2	49	394	493
	Died	0	0	0	0	0
	Euthanasia	1	1	17	4	11
	Return to owner	213	2	67	415	138
	Transfer	100	37	189	118	1216
	Errore classe (%)	14.32	100	94.72	55.42	34.55

Tabella 6: Errori di classificazione con GAM.

La Tabella 7 riporta gli errori di classificazione commessi dal modello, per i quali valgono le stesse considerazioni fatte per la regressione logistica, alle quali si aggiunge il fatto che anche la classe *Euthanasia* non viene mai assegnata.

Questo fatto può essere osservato andando ad effettuare il `plot` dell'albero ottenuto, dal quale è possibile osservare come nessuna foglia dell'albero abbia etichetta *Died* o *Euthanasia*[4].

C'è però da tenere in considerazione che questo modello è stato calcolato senza utilizzare le variabili `PrimaryBreed` e `SecondaryBreed` perché con la funzione `tree` non è possibile usare variabili di tipo `Factor` con più di 32 livelli ed entrambe le variabili sforavano questo limite. Nonostante ciò il punteggio ottenuto su Kaggle è di 0.87657, ovvero migliore rispetto a quello ottenuto dalla regressione logistica.

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1765	2	33	352	431
	Died	0	0	0	0	0
	Euthanasia	0	0	0	0	0
	Return to owner	289	3	86	465	206
	Transfer	138	37	203	114	1221
	Errore classe (%)	19.47	100	100	52.05	34.28

Tabella 7: Errori di classificazione con l'albero di classificazione.

3.5 Random Forest

Un miglioramento al modello degli alberi di classificazione è dato dal modello delle random forest, le quali effettuano la classificazione combinando più alberi di dimensioni ridotte.

Trattandosi di una combinazione di alberi di classificazione, anche questo modello soffre del problema relativo alle variabili `PrimaryBreed` e `SecondaryBreed`, le quali sono state scartate dal dataset.

I parametri del modello che si sono rivelati essere migliori sono mtry uguale a 12 e `ntree` uguale a 600, ovvero la configurazione migliore della foresta ha 600 alberi di classificazione, addestrati su un set ridotti di osservazioni utilizzando tutte e 12 le variabili a disposizione. Infatti, il parametro `mtry` specifica il numero di variabili da scegliere a caso tra quelle disponibili quando viene estratto un campione per calcolare un singolo albero. Gli altri valori provati per `mtry` sono stati 5 e 10, ma hanno prodotto risultati peggiori.

Con questa configurazione, il modello ha ottenuto un errore di classificazione del 18.63% sul set di validazione, tuttavia il punteggio calcolato da Kaggle risulta essere 2.1834.

Come prima causa di questa discrepanza si è pensato all'*overfitting* dei dati da parte della foresta. Tuttavia, le due metriche sono state calcolate su una porzione del dataset che non è stata utilizzata per calcolare il modello e quindi l'overfitting dovrebbe aver fatto aumentare anche l'errore di classificazione sul set di validazione.

Un'altra possibile causa deriva dal fatto che Kaggle sembra penalizzare maggiormente le previsioni certe, ovvero quello che danno probabilità 1 ad una determinata classe e si è notato che il modello ha effettuato circa il 10% di previsione certe per le classi *Adoption* e *Transfer*.

3.6 Reti neurali

Come modello di una rete neurale è stato utilizzato quello disponibile nel pacchetto `nnet`, il quale permette di impostare alcuni iper-parametri come:

[4]Il plot dell'albero è stato omesso perché è troppo grande rispetto alle dimensioni del documento

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1961	3	27	224	259
	Died	0	29	0	0	2
	Euthanasia	4	1	203	7	15
	Return to owner	141	2	29	646	72
	Transfer	86	7	63	54	1510
	Errore classe (%)	10.53	30.95	36.95	30.61	18.72

Tabella 8: Errori di classificazione con random forest.

- `MaxNWts`: massimo numero di pesi nella rete. Impostato a 10000 perché il valore di default è troppo basso rispetto al numero di variabili.
- `maxit`: massimo numero di iterazioni nell'addestramento della rete. Durante l'ottimizzazione dei parametri è stato mantenuto fissato a 200, mentre per i due modelli finali è stato portato a 1000.
- `decay`: fattore di decadimento dei pesi per l'algoritmo di addestramento. Impostato a 0.001668, valore individuato utilizzando i dati del *train* e *test set*.
- `size`: numero di nodi presenti nello strato nascosto della rete. Impostato a 10 dal momento che non c'è un modo sistematico per stabilire il numero ottimale ed effettuare l'ottimizzazione di questo iper-parametro richiede troppo tempo.

La rete così ottenuta ha prodotto un errore di classificazione del 29, 31%, senza mai prevedere la morte naturale per gli animali, e un punteggio di 0.93989 su Kaggle. La Tabella 9 riporta più in dettaglio gli errori commessi dalla rete.

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1892	4	41	338	375
	Died	0	0	0	0	0
	Euthanasia	2	1	63	8	20
	Return to owner	188	1	70	495	135
	Transfer	110	36	148	90	1328
	Errore classe (%)	13.68	100	80.43	46.83	28.52

Tabella 9: Errori di classificazione con la rete neurale.

3.7 Boosting

Come modello di Boosting è stata utilizzata la versione di default della libreria `ada` e dal momento che con questo modello non è possibile effettuare la classificazione multiclasse, è stato necessario adottare la stessa strategia utilizzata per il GAM e per la regressione logistica.

Sono stati quindi calcolati 5 modelli, uno per ogni possibile valore della variabile risposta e combinati tra loro per produrre la previsione finale.

Le matrici di confusione dei vari modelli sono riportate nella Tabella 10 mentre la Tabella 11 contiene gli errori di classificazione.

Dalla matrice di confusione del modello `OutDied` si può notare che, anche se non viene mai prevista la classe *Died*, l'errore commesso è minimo e questo è causato dal fatto che nel dataset completo, le osservazioni con questa classe sono in netta minoranza rispetto le altre.

L'errore totale commesso dal modello è del 32, 61% e il punteggio ottenuto su Kaggle è di 0.88321.

		Valori predetti	
		No	Yes
Valori osservati	No	2494	659
	Yes	495	1697

(a) OutAdoption, errore: 0.215

		Valori predetti	
		No	Yes
Valori osservati	No	5303	0
	Yes	42	0

(b) OutDied, errore: 0.007

		Valori predetti	
		No	Yes
Valori osservati	No	5019	4
	Yes	302	20

(c) OutEuthanasia, errore: 0.057

		Valori predetti	
		No	Yes
Valori osservati	No	4293	121
	Yes	752	179

(d) OutReturnToOwner, errore: 0.163

		Valori predetti	
		No	Yes
Valori osservati	No	3220	267
	Yes	763	1095

(e) OutTransfer, errore: 0.192

Tabella 10: Matrici di confusione per i 5 classificatori con Boosting

		Valori osservati				
		Adoption	Died	Euthanasia	Return to owner	Transfer
Valori predetti	Adoption	1900	4	53	393	475
	Died	0	0	0	0	0
	Euthanasia	1	0	17	3	4
	Return to owner	185	1	70	437	131
	Transfer	106	37	182	98	1248
	Errore classe (%)	13.32	100	94.72	53.06	32.83

Tabella 11: Errori di classificazione con Boosting.

4 Conclusioni

4.1 Riepilogo

I risultati ottenuti dai vari modelli sono riportati in Tabella 12 e come si può notare, il modello migliore che ha ottenuto un errore minore è quello delle random forest, anche se il punteggio che ha ottenuto su Kaggle è quello peggiore. Il miglior modello secondo Kaggle è il GAM, che sulla classifica provvisoria del sito si trova in posizione 480 su 876.

4.2 Considerazioni aggiuntive

- L'estrazione delle informazioni del dataset si è basata su considerazioni generali e su alcune semplici ricerche, risulta quindi limitata, specialmente per quanto riguarda la razza degli animali e il colore del pelo. È infatti ragionevole pensare che ci siano razze più rare, che possono venire adottate più facilmente rispetto ad altre e che certe combinazioni di colori

Modello	Variante	Errore di classificazione (%)	Punteggio su Kaggle
GLM	`multinom`	33.66	0.89358
	`glm`	39.86	0.87527
MARS	-	36.40	0.90010
GAM	-	34.04	0.87405
Albero	-	35.43	0.87657
Random Forest	-	18.63	2.1834
Rete neurale	-	29.31	0.93989
Boosting	-	32.61	0.88321

Tabella 12: Riepilogo dei risultati ottenuti con i vari modelli.

siano più popolari rispetto ad altre, oppure che la razza influisca sulle dimensioni e sul carattere dell'animale, fattori che possono essere importanti per decidere se adottare o meno un animale.

- Il dataset contiene poche osservazioni classificate come *Died* (0.73%), potrebbe essere conveniente calcolare un modello dedicato a predirre questa classe, utilizzando una porzione più bilanciata del dataset.
- Per la regressione logistica e per il GAM sarebbe necessario effettuare una selezione *stepwise* delle variabili più significative in modo da ottenere un modello migliore. Tuttavia, per motivi di tempo e dato che la selezione *stepwise* sarebbe stata necessaria per ognuno dei modelli binomiali, questa non è stata effettuata.
- Per come sono definiti gli alberi su R, non è stato possibile utilizzare le variabili `PrimaryBreed` e `SecondaryBreed` nel calcolo del modello. Uno modo per includerle potrebbe essere quello di scomporre le variabili in ulteriori variabili, ognuna con meno di 32 possibili valori. Questa suddivisione non è stata effettuata perché ritenuta poco utile.
- Quando non è stato possibile trovare un classificatore multiclasse sono stati combinati più classificatori binomiali secondo la strategia *one-vs-all*. Un'alternativa, che non è stata provata, è quella di utilizzare la strategia *one-vs-one* che si basa su un numero più elevato di classificatori, ognuno dei quali deve scegliere tra due possibili classi anziché scegliere se un'osservazione è di una determinata classe o meno.