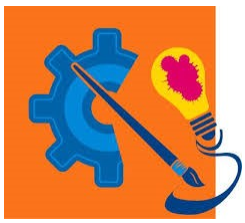


ISTITUTO TECNICO INDUSTRIALE E.FERMI

INDIRIZZO INFORMATICA E TELECOMUNICAZIONI

F.Nicotra, A.Resta, G.Stagnitta

Classe: V

Sezione: A

Indirizzo: Informatica

Elaborazione dei dati per il riconoscimento del mezzo partendo dai sensori dello smartphone

---

Relazione Alternanza Scuola-Lavoro

ANNO SCOLASTICO 2018 / 2019

Capitolo 1. Introduzione

Gli obiettivi del progetto Eco-Logic CT sono finalizzati alla raccolta di informazioni sul comportamento di mobilità degli studenti quando si recano in una o più sedi universitarie.

I dati raccolti vengono elaborati al fine di promuovere la riduzione dell'emissione di gas serra causato dalla mobilità di studenti e personale dell'Ateneo derivante da un uso eccessivo ed inefficiente dell'auto privata.

Le attività che compongono questo progetto le possiamo raggruppare in tre grandi fasi principali:

- Acquisizione dei dati provenienti dai sensori dello smartphone
- Elaborazione dei dati mediante algoritmi di machine learning
- Visualizzazione dei risultati

In questa relazione è dettagliata ed attenzionata solo la parte di elaborazione dei dati, che ha coinvolto il team della scuola I.T.I.S Enrico Fermi di Giarre.

Capitolo 2. Applicazione mobile

2.1. Descrizione

La prima fase consiste nel creare un'applicazione per smartphone in grado di leggere e registrare i dati provenienti da alcuni sensori del telefono come accelerometro, giroscopio, magnetometro e così via. Le letture avvengono ad una frequenza di 20 Hz, quindi in un arco temporale di 5 secondi avremo circa 100 letture.

I dati raccolti andranno conservati in un database relazionale MySQL. Successivamente saranno elaborati, al fine di rilevare il modo di trasporto utilizzato dallo studente per andare all'Ateneo.

2.2. Utilizzo dei sensori

Lo studente fornirà -attraverso l'applicazione- il dato riguardante la modalità di trasporto utilizzata per andare all'università. Per la validazione di quanto inserito ricorriamo alla *Transport Mode Detection,* descritta più avanti quando parleremo della fase elaborativa.

In questo prototipo sono stati sfruttati i soli dati del sensore accelerometro, che hanno prodotto ottimi risultati, ma se vogliamo un'accuratezza maggiore nella predizione del mezzo è opportuno aggiungere al dataset i dati provenienti da più sensori.

IDMATRICOLA	VIAGGIO	DATA	TIPO	X	Y	Z	TIMESTAMP
1	1	2019-03-03	Auto	0.19153612852097	9.3080577850342	3.2650926113129	1551630437890
1	1	2019-03-03	Auto	0.019153613597155	9.2505970001221	2.4797945022583	1551630437970
1	1	2019-03-03	Auto	0.057460840791464	9.3655185699463	3.3321301937103	1551630438036
1	1	2019-03-03	Auto	0.3687070608139	9.2697505950928	2.7814638614655	1551630438106
1	1	2019-03-03	Auto	0.50278234481812	9.3607301712036	2.9777884483337	1551630438177
1	1	2019-03-03	Auto	0.24420857429504	9.3894605636597	2.9251158237457	1551630438209
1	1	2019-03-03	Auto	0.24420857429504	9.3894605636597	2.9251158237457	1551630438209
1	1	2019-03-03	Auto	1.1779472827911	9.0925798416138	3.4087445735931	1551630438328
1	1	2019-03-03	Auto	1.1779472827911	9.0925798416138	3.4087445735931	1551630438328
1	1	2019-03-03	Auto	1.0917559862137	8.7142953872681	-4.7411179542542	1551630438429
1	1	2019-03-03	Auto	1.039083480835	9.6528224945068	-0.19213469326496	1551630438495
1	1	2019-03-03	Auto	0.28251579403877	10.792462348938	9.5043821334839	1551630438508
1	1	2019-03-03	Auto	2.3654713630676	7.9720931053162	9.7821092605591	1551630438568
1	1	2019-03-03	Auto	6.0381765365601	6.3440361022949	0.4064157307148	1551630438628
1	1	2019-03-03	Auto	7.2113356590271	6.6983776092529	10.90259552002	1551630438669
1	1	2019-03-03	Auto	7.7045412063599	5.5300073623657	4.0360255241394	1551630438728

Esempio di dati "grezzi" dell'accelerometro conservati in una tabella del database

Capitolo 3. Elaborazione dati

3.1. Introduzione

La fase di elaborazione dei dati ha come fine quello di effettuare le *predictions* della modalità di trasporto, basandosi sui dati strutturati del database creato nella parte applicativa.

Il linguaggio di programmazione adottato è il Python e la libreria che ci ha permesso di sfruttare il machine learning (descritto più avanti) è la *scikit-learn*.

Nel dettaglio questa fase la possiamo scomporre in 4 grandi attività:

- Creazione del dataset
- Ricerca dell'algoritmo
- Creazione del modello
- Predictions

Ognuna di queste è uno script Python.

3.2. Machine learning

Il machine learning è una branca dell'informatica basata su algoritmi che apprendono automaticamente (Intelligenza artificiale).

Esso "*esplora lo studio e la costruzione di algoritmi che possono imparare dai dati*"(Dott. Luca Naso), forniti sotto forma di dataset.

Questi dati vengono visti dall'algoritmo utilizzato come "esempi". All'aumentare del numero di esempi aumenteranno le prestazioni in termini di precisione delle *predictions*.

Gli algoritmi usano metodi matematico-computazionali per apprendere direttamente dai dati senza equazioni o modelli predeterminati, risolvendo problemi di cui non conosciamo la soluzione.

Esistono due tipi di apprendimento:

- Supervisionato, dove sono presenti dati di input e dati di output etichettati
- Non supervisionato, dove invece sono presenti solo dati di input

Mentre i tipi di output si suddividono in:

- Regressione, in cui l'output è un valore continuo
- Classificazione, in cui l'output è un valore categorico o discreto
- Clustering, dove avviene il raggruppamento dei dati di input

3.2.1. Python

Python è un linguaggio di programmazione ad alto livello, in cui è possibile sviluppare applicazioni distribuite, di scripting e molte altre. Lo abbiamo scelto proprio per la sua versatilità e semplicità.

Sono disponibili, inoltre, molte librerie come pandas (per le strutture dati), math (calcolo delle caratteristiche), matplotlib (illustrazioni grafiche) e scikit-learn (ML e classificatori).

L'ambiente di sviluppo adottato è stato Pycharm di JetBrains.

3.3. Creazione del dataset

Il dataset viene creato sulla base dei valori provenienti dai sensori dello smartphone che stiamo considerando. Per questo progetto è stato utilizzato un solo sensore: l'accelerometro. Tuttavia per una maggiore fedeltà nelle *predictions* è possibile "arricchire" il dataset con sensori significativi quali giroscopio, magnetometro e pressione (per riconoscere i viaggi in treno).

Dai dati grezzi dell'accelerometro (X, Y, e Z) viene ricavato un solo valore secondo la formula "*sqrt(X^2+Y^2+Z^2)*", ottenendo l'accelerazione assoluta.

Una volta fatta quest'operazione il dataset verrà diviso secondo finestre temporali di 5 secondi, sulle quali calcoleremo le caratteristiche: Media, Massimo, Minimo e Deviazione standard.

In ogni finestra ci saranno un totale di circa 100 letture, poiché esse vengono registrate dai sensori ad una frequenza di 20 Hz, periodicamente durante il tragitto.

	IDMATRICOLA	VIAGGIO	TARGET	MEDIA ACCELEROMETRO	MINIMO ACCELEROMETRO	MASSIMO ACCELEROMETRO	DEV STD ACCELEROMETRO
0	1	1	Car	10.300699528915729	4.407808661644344	20.6450525415675	2.676375194571995
1	1	1	Car	10.195394981879394	4.407808661644344	20.6450525415675	2.0015820195298715
2	1	1	Car	10.055651620727641	7.695597901667265	12.02728754488944	0.9246310956480974
3	1	1	Car	9.76368025248263	7.6458794212955175	12.778532895992376	1.0468786506691463
4	1	1	Car	9.921245078376172	7.6458794212955175	12.778532895992376	1.1292824473310945
5	1	1	Car	10.105831918276692	6.158699140275044	13.117337056435378	1.172683607828895
6	1	1	Car	10.347503202886083	7.289580106152708	12.213871028394111	0.8869365847478193
7	1	1	Car	10.061603086560378	7.503952879135095	13.368963719437051	1.0870879560693145
8	1	1	Car	9.853165084075759	6.120983377052467	13.311879553055181	1.227888561118259
9	1	1	Car	9.773427946415675	5.767864389603281	14.317809127357583	1.5470672493138318
10	1	1	Car	10.197372685933356	7.775828139355862	13.803575436312412	1.2137536881213475
11	1	1	Car	10.120205067844397	8.306539139266352	12.312047416408294	0.9212661542375025
12	1	1	Car	10.064548721219559	8.790661055719248	12.150538708692745	0.7157873545797844
13	1	1	Car	10.067388173374145	8.223908221911651	12.37708490227624	0.817126995794025
14	1	1	Car	10.189852775494382	6.671093351276652	15.895048390423954	1.2544269291469996
15	1	1	Car	10.042808227061988	7.245672290047664	13.84085775343311	1.4363593462395436

Esempio di dataset con caratteristiche dell'accelerometro post-elaborazione

3.4. Normalizzazione

La normalizzazione consiste nel portare tutti i dati del dataset in un range compreso tra "0" e "1". Viene effettuata per eliminare alcuni "fattori di disturbo" come valori aventi ordine di grandezza superiore rispetto ad altri. Il classificatore potrebbe vedere le *features,* contenenti questi valori, come "più importanti" e dunque influenzare negativamente la creazione del modello.

Alcuni classificatori come il Random Forest non sono influenzati dalla normalizzazione del dataset, di conseguenza questa sarà irrilevante.

3.5. Ricerca dell'algoritmo

La ricerca dell'algoritmo serve a conoscere il classificatore più adatto al nostro dataset, ovvero quello che restituisce in output un'accuratezza nella predizione del modo di trasporto maggiore rispetto ad altri.

Il classificatore che abbiamo deciso di utilizzare per questo progetto è stato il Random Forest che con i soli dati dell'accelerometro ci ha dato un'accuratezza del 70%.

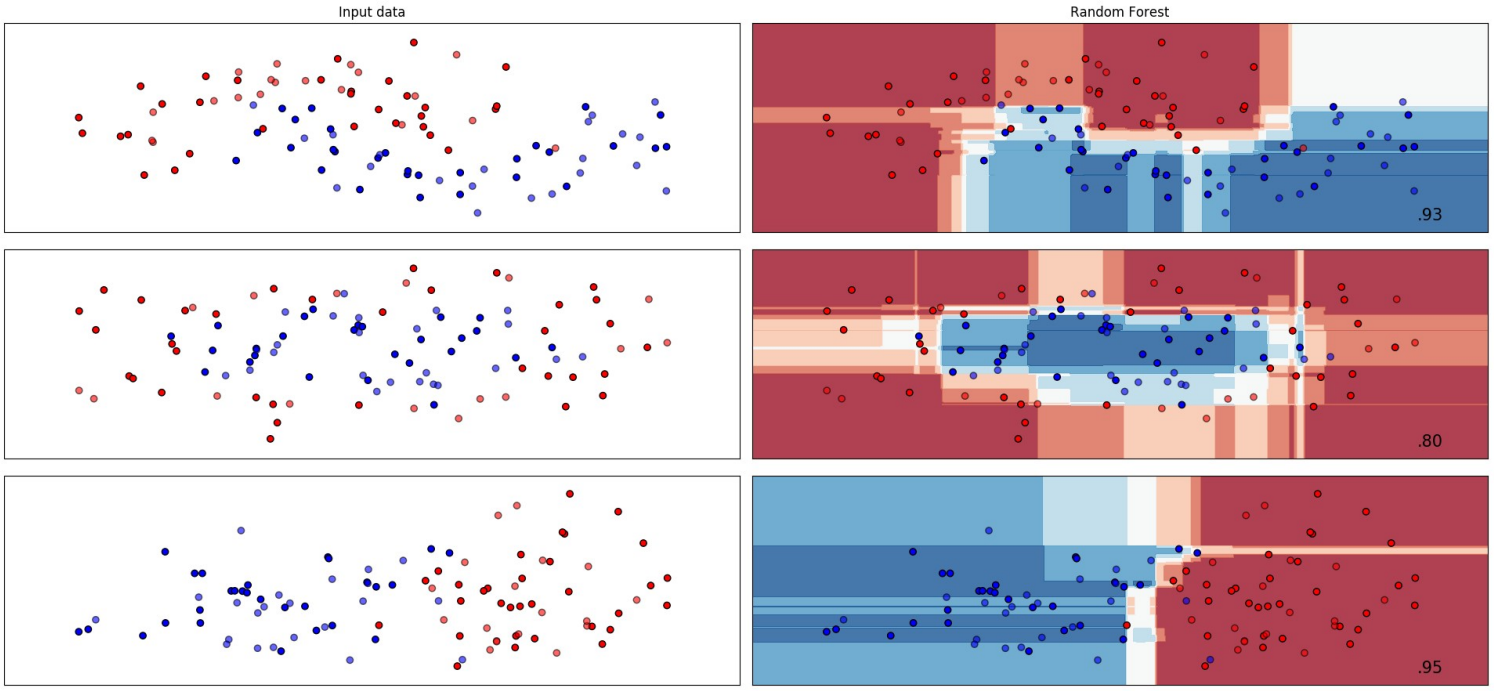
Un'altro classificatore emerso dalla ricerca che ha restituito un'accuratezza di circa 67% è stato il Gaussian Process, che al contrario del Random Forest è influenzato dalla normalizzazione.

In questa ricerca sono stati analizzati 9 classificatori:

- KNeighbors
- SVC (ripetuto due volte con parametri diversi)
- Gaussian Process
- Decision Tree
- Random Forest
- MLP
- AdaBoost
- GaussianNB
- Quadratic Discriminant Analysis

Nel Random Forest si deve inserire, tramite parametro, il numero di estimatori. Questo intero indica il numero di sotto-insiemi creati dal dataset originario. Ciascun sotto-insieme o albero deve restituire una classificazione, dati i soli valori di input.

Sono state effettuate delle prove variando il parametro "*n_estimators*", ed è emerso che 100 (il massimo) genera un'accuratezza maggiore, per quanto riguarda il nostro dataset.



L’immagine soprastante rappresenta alcuni raggruppamenti effettuati dal Random Forest partendo da un set mischiato di valori di input (*features*). Ogni colore identifica una modalità di trasporto diversa.

3.6. Creazione del modello

In questo processo i classificatori calcolano, dal dataset, un modello, utilizzato per dividere in classi le nuove istanze.

Esempio di codice per allocazione ed addestramento del modello:

```
from sklearn import ensemble
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20)
model = ensemble.RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

Per effettuare il testing sullo stesso dataset di training abbiamo splittato il dataset con proporzione 80% training e 20% test, in questo modo una parte è riservata all'addestramento ed un'altra ai test.

La terza riga di codice mostra l'allocazione del modello con il parametro "*n_estimator*=100", questo significa che il Random Forest utilizzerà 100 sotto-insiemi per le *predictions*.

Per l'allenamento utilizziamo il metodo *fit()* parametrizzato dai valori delle caratteristiche (*X_train*) ed i valori del target o mezzo (*y_train*).

X_train contiene i valori di Media, Minimo, Massimo e Deviazione standard della porzione di dataset "1 – test_size".

y_train contiene i target (mezzi) corrispondenti ai valori di *X_train*.

Sono state effettuate delle prove con diverse proporzioni, ed il risultato è pressoché simile, ma aumentando di molto il "*test_size*" l'accuratezza diminuisce. Questo avviene perché il modello non è allenato abbastanza.

3.6.1. Model persistence

Essendo che l'addestramento del modello avviene una sola volta, questo si deve salvare su disco per poterlo riutilizzare, aprendolo in "*rb*" su un nuovo script senza ripetere il *training*. Da questo processo, chiamato *model persistence*, ne deriva un file che contiene il modello già allenato con i dati specifici di quel dataset, ma se dobbiamo apportare modifiche sui target oppure dobbiamo utilizzare il modello su un'altra città (vedi cap. 4), che quindi avrà un diverso dataset, potremmo doverlo ri-allenare.

La libreria utilizzata per il *model persistence* è *pickle.*

3.7. Predictions

L'attività di *prediction* è lo step finale della fase elaborativa e serve per validare il modo di trasporto inserito dall'utente nell'applicazione. Essa consiste nell'utilizzare il metodo *predict()* messo a disposizione dalla *scikit-learn* per effettuare delle predizioni avendo soltanto i valori delle caratteristiche. Questo metodo mette in funzione il nostro classificatore, che deve restituirci un risultato.

Dal focus sul machine learning è possibile intuire che questo tipo di output è un valore categorico, mentre l'apprendimento è supervisionato perché sono presenti sia dati di input che dati di output.

Il modello utilizzato, ricordiamo, sarà quello precedentemente salvato.

```
predicted_test = model.predict(X_test)
```

Per misurare l'accuratezza della *prediction*, ovvero la percentuale di classificazioni corrette, la *scikit-learn* fornisce il metodo *accuracy_score()*.

```
from sklearn.metrics import accuracy_score
print('Test score')
print(accuracy_score(y_test, predicted_test))
```

Come parametri dobbiamo passare sia i dati di test predetti, sia i corrispettivi target reali.

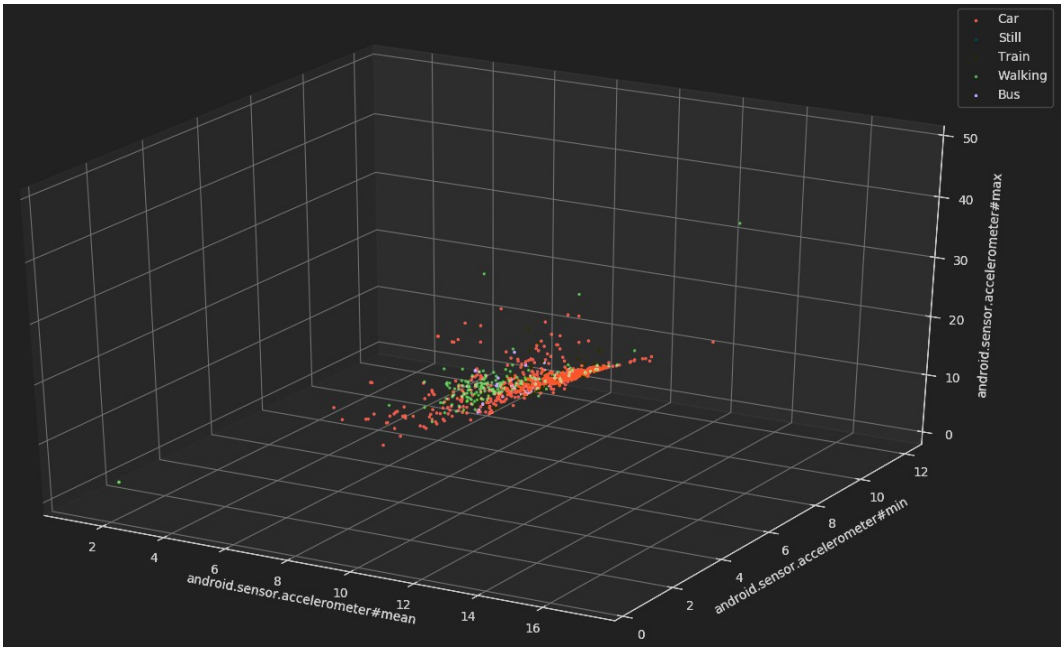
3.8. Grafico 3D delle features

In figura è riportata una rappresentazione grafica delle *features* Media, Minimo e Massimo nello spazio. I diversi colori indicano rispettivamente i vari mezzi di trasporto (Auto, Fermo, Treno, A piedi e Bus).

L'attività che fanno i classificatori è quella di raccogliere tutti i punti appartenenti ad uno stesso target, creando tanti gruppi o insiemi. In questo caso i punti risultano molto mischiati e poco separabili, di conseguenza le *predictions* non saranno molto affidabili.

L'utilizzo di più sensori porta ad una maggiore separabilità e dunque ad una accuratezza migliore.

Ogni classificatore ha un diverso modo di separare le caratteristiche, che potrebbe essere o meno quello adatto al tipo di dataset che abbiamo; per questo motivo la ricerca dell'algoritmo è di fondamentale importanza.



Rappresentazione nello spazio delle features

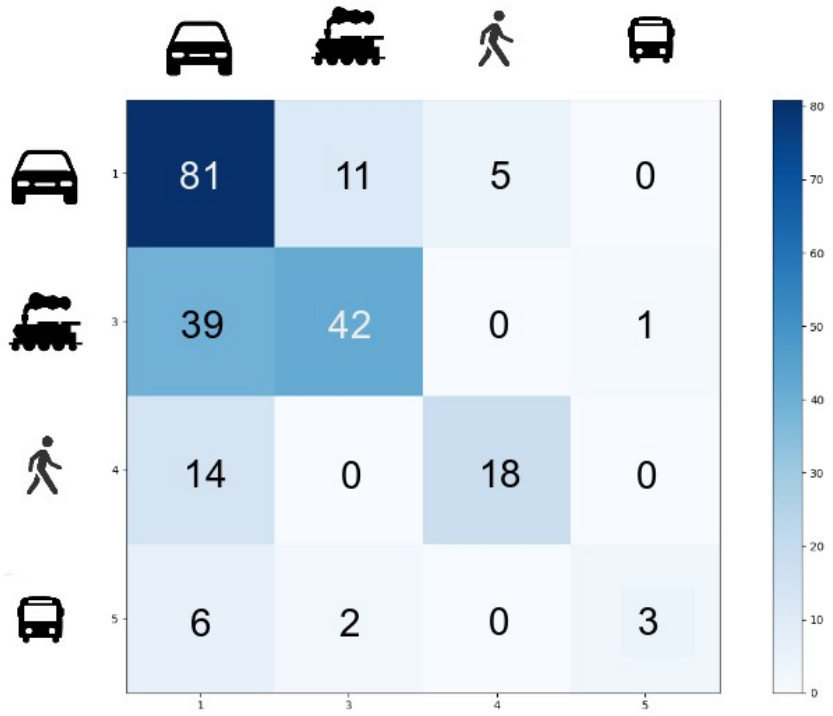
3.9. Matrice di confusione

Per avere un quadro più dettagliato delle *predictions* occorre introdurre un nuovo strumento, la matrice di confusione. Anch'essa viene fornita da *scikit-learn* ed indica il numero di punti per i quali il target previsto è uguale al target reale.

Questo *tool* è molto utile soprattutto ai programmatori per vedere dove il modello può essere migliorato, permettendo di analizzare le *predictions* errate (per es. come le modalità di trasporto sono state etichettate nei risultati).

Le colonne della matrice di confusione rappresentano i valori predetti, mentre le righe rappresentano i valori reali.

La diagonale mostra le letture correttamente etichettate dal classificatore, mentre il resto mostra il numero di classificazione errate.



Esempio di matrice di confusione

Nell'esempio in foto il numero "81" della prima riga indica il numero di letture correttamente classificate, mentre "11", "5" e "0" indicano il numero di letture in cui il target reale (Auto) viene confuso rispettivamente con Treno, A piedi e Bus.

```
From sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, predicted_test))
print(classification_report(y_test, predicted_test))
```

```
[[83 18  4  1]
 [31 45  2  1]
 [ 8  0 16  0]
 [ 2  8  0  3]]
```

	precision	recall	f1-score	support
1	0.67	0.78	0.72	106
3	0.63	0.57	0.60	79
4	0.73	0.67	0.70	24
5	0.60	0.23	0.33	13

Così come nell'*accuracy_score()* anche nella *confusion_matrix()* dobbiamo passare come parametri i dati predetti ed i target.

Seguito dalla matrice viene graficato anche il "report", che arricchisce di informazioni aggiuntive la *prediction*:

- Precision, che rappresenta la percentuale di classificazioni positive che sono corrette.

- Recall, che rappresenta invece la percentuale di classificazioni positive del testing set, comprendendo anche i falsi positivi.

Capitolo 4. Flessibilità della soluzione

4.1. Acquisizione nuovi sensori

L'acquisizione di nuovi sensori genererebbe nel dataset nuovi valori che, aggiunti ai precedenti, aumenterebbero il loro grado di separabilità, apportando migliorie in termini di fedeltà nelle predizioni del modo di trasporto.

Nello studio effettuato dall'Università di Bologna sono stati utilizzati 9 sensori, ottenendo, con il Random Forest, un'accuratezza del 96%. Da questo possiamo notare che più dati, da parte di vari sensori, si raccolgono durante i viaggi, più la probabilità di ottenere la classificazione corretta aumenta.

Considerando che gli smartphone non presentano tutti gli stessi sensori per l'acquisizione dei dati, si deve scegliere attentamente come e quali di questi utilizzare. Ad esempio l'accelerometro ed il giroscopio sono ormai sempre presenti.

4.2. Replicare il modello su altre città

Con il *model persistence* abbiamo salvato il modello allenato tramite il dataset di Catania, ma se vogliamo che le *predictions* siano relative ad altre città possiamo ri-allenarlo con un diverso dataset, che avrà le stesse caratteristiche di quello di Catania (Media, Minimo, Massimo e Deviazione standard) calcolate però sulla base di valori diversi. Ad esempio un viaggio in autobus produrrà in output valori che differiscono a seconda delle città in cui ci troviamo, per motivi come l'intensità del traffico, la struttura delle strade ecc.

4.3. Variazione dei target

I mezzi di trasporto di cui abbiamo raccolto i dati si sono limitati ad Autobus, Treno, Macchina ed A piedi, ma è possibile aggiungerne molti altri con gli stessi scripts Python. Ad esempio è possibile aggiungere mezzi come biciclette o motocicli.

Questa variazione non influisce sulla funzionalità dell'elaborazione, proprio perchè la nostra soluzione è flessibile.

4.4. Altri sviluppi futuri: Feedback

Dopo l'arrivo a destinazione dello studente si potrebbe pensare di inviargli un modulo di feedback. L'informazione richiesta sarà facoltativa e personalizzata in base alla modalità utilizzata per spostarsi. Ciò che si vuole conoscere è l'esperienza del viaggio. Un esempio è portato di seguito:

- Metro/Autobus:
  - Indice di gradimento del servizio
  - Ritardi/Anticipi dei mezzi
- Mezzo privato:
  - Indice dell'intensità del traffico
  - Condizioni meteorologiche

I feedback avranno due fini importanti:

- mettere a conoscenza delle situazioni sui vari mezzi di trasporto gli altri studenti dell'università;

- segnalare i problemi riscontrati durante il viaggio dovuti a disservizi dei trasporti pubblici.

Capitolo 5. Conclusioni

Durante il periodo di alternanza scuola-lavoro si è lavorato per la creazione di un prototipo di sistema che possa essere impiegato nel campo della *Transport Mode Detection*. Esso apre le porte allo studio delle metodologie di trasporto usate dall'Ateneo dell'Università di Catania.

La conoscenza dei principi di Machine Learning è stata propedeutica allo sviluppo del software.

Questa esperienza ci ha esposto ad una tematica tutt'oggi in fase di sviluppo e perfezionamento. Abbiamo avuto la possibilità di mettere le mani su qualcosa di nuovo e non presente in un normale programma scolastico. Il lavoro svolto è stato ulteriormente apprezzato sapendo di aver contribuito al territorio di cui facciamo parte.

Appendice

Appendice A. Codice sorgente: creazione del dataset

```
import pandas
import numpy
import math
#Lettura del dataset contenente i valori grezzi dell'accelerometro
df = pandas.read_csv('accelerazioni.txt', sep=';', engine='python')   #nrows=20
df.sort_values(by=['DATA', 'IDMATRICOLA', 'VIAGGIO', 'TIMESTAMP'])
col_names =  ['IDMATRICOLA', 'VIAGGIO', 'TIPO', 'debugACC',
              'MEDIA ACCELEROMETRO',
              'MINIMO ACCELEROMETRO',
              'MASSIMO ACCELEROMETRO',
              'DEV STD ACCELEROMETRO',
              'numletture', 'diffTS']
new_df  = pandas.DataFrame(columns = col_names)
#primo record
oldmatricola = df.loc[0,'IDMATRICOLA']
oldviaggio = df.loc[0,'VIAGGIO']
oldtimestamp = df.loc[0,'TIMESTAMP']
ACC = math.sqrt(df.loc[0,'accX']**2 + df.loc[0,'accY']**2 + df.loc[0,'accZ']**2)
numletture=1
npACC=[ACC]
#si parte dal secondo
for i in range(1, len(df)):
    currmatricola = df.loc[i, 'IDMATRICOLA']
    currviaggio = df.loc[i, 'VIAGGIO']
    currtimestamp = df.loc[i, 'TIMESTAMP']
                currACC=math.sqrt(df.loc[i,'accX']**2    +    df.loc[i,'accY']**2    +
df.loc[i,'accZ']**2)
    if ((currmatricola != oldmatricola) or
        (currviaggio != oldviaggio) or
        (currtimestamp > oldtimestamp + 5000) or
        (i==len(df)-1)):
        #nuovo record dataset
        ACC=ACC/numletture          #debug
        mynp = numpy.array(npACC)
        new_df.loc[len(new_df)] = [oldmatricola,
                                   oldviaggio,
                                   df.loc[i-1, 'TIPO'],
                                   ACC,
                                   mynp.mean(),
                                   mynp.min(),
                                   mynp.max(),
                                   mynp.std(),
                                   numletture,
                                   df.loc[i-1, 'TIMESTAMP'] - oldtimestamp]
        oldmatricola=currmatricola
        oldviaggio=currviaggio
        oldtimestamp=currtimestamp
        numletture=1                #debug
        ACC=currACC                 #debug
        npACC=[currACC]
    else:
        #stesso record
        numletture = numletture+1   #debug
        ACC = ACC + currACC         #debug
        npACC.append(currACC)
new_df.to_csv('dataset.txt', sep=';')
```

Appendice B. Codice sorgente: ricerca dei classificatori

```
from sklearn.neural_network import MLPClassifier


from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas
import numpy
from numpy import zeros
df = pandas.read_csv('dataset.txt', sep=";")
df = df.loc[:, ['MEDIA ACCELEROMETRO',
                'MINIMO ACCELEROMETRO',
                'MASSIMO ACCELEROMETRO',
                'DEV STD ACCELEROMETRO',
                 'TIPO']]
#Creazione del vettore "y" contenente i target sotto forma di numeri
y = zeros((len(df),), dtype=numpy.int)
for i in range(0, len(df)):
    if df.iloc[i]['TIPO'] == 'Auto':
        y[i] = 1
    if df.iloc[i]['TIPO'] == 'Fermo':
        y[i] = 2
    if df.iloc[i]['TIPO'] == 'Treno':
        y[i] = 3
    if df.iloc[i]['TIPO'] == 'A piedi':
        y[i] = 4
    if df.iloc[i]['TIPO'] == 'Bus':
        y[i] = 5
    if df.iloc[i]['TIPO'] == 'Bici':
        y[i] = 6
#Rimpiazzo i valori nulli del dataset con la media del df
replace_with = df.mean()
df = df.fillna(replace_with)
#Conservo in "X" i valori delle 4 caratteristiche del df
X = df.loc[:, ['MEDIA ACCELEROMETRO',
               'MINIMO ACCELEROMETRO',
               'MASSIMO ACCELEROMETRO',
               'DEV STD ACCELEROMETRO'
               ]].values
#Divido il dataset per ottenere 80% train e 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
#Vettore contenente i nomi delle caratteristiche (usato nella normalizzazione)
features=['MEDIA ACCELEROMETRO',
          'MINIMO ACCELEROMETRO',
          'MASSIMO ACCELEROMETRO',
          'DEV STD ACCELEROMETRO'
          ]
#normalizzazione
df_norm = df.copy()
to_norm = df[features]
df_norm[features] = (to_norm - to_norm.min())/(to_norm.max()-to_norm.min())
df = df_norm
#vettore di classificatori per testare i vari algoritmi
classifiers = [
    KNeighborsClassifier(3),
```

```
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    GaussianProcessClassifier(1.0 * RBF(1.0)),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=100),
    MLPClassifier(alpha=1),
    AdaBoostClassifier(),
    GaussianNB(),
    QuadraticDiscriminantAnalysis()]
class_len= len(classifiers)
a=[]
accuracy=0
#Effettuo 10 predictions per ogni classificatore
for i in range(class_len):
    model = classifiers[i]
    model.fit(X_train, y_train)
    for j in range (10):
        predicted_test  = model.predict(X_test)
        accuracy= accuracy+accuracy_score(y_test, predicted_test)
#Salvo la media delle accuratezze nel vettore “a”
    a.append(accuracy/10)
    accuracy=0
print(a)
```

Appendice C. Codice sorgente: creazione del modello

```
from sklearn import ensemble
import pickle
import pandas
import numpy
from numpy import zeros
df = pandas.read_csv('dataset.txt', sep=";")
df = df.loc[:, ['MEDIA ACCELEROMETRO',
                'MINIMO ACCELEROMETRO',
                'MASSIMO ACCELEROMETRO',
                'DEV STD ACCELEROMETRO',
                 'TIPO']]
y = zeros((len(df),), dtype=numpy.int)
for i in range(0, len(df)):
    if df.iloc[i]['TIPO']=='Auto':
        y[i] = 1
    if df.iloc[i]['TIPO'] == 'Fermo':
        y[i] = 2
    if df.iloc[i]['TIPO'] == 'Treno':
        y[i] = 3
    if df.iloc[i]['TIPO'] == 'A piedi':
        y[i] = 4
    if df.iloc[i]['TIPO'] == 'Bus':
        y[i] = 5
    if df.iloc[i]['TIPO'] == 'Bici':
        y[i] = 6
replace_with = df.mean()
df = df.fillna(replace_with)
X = df.loc[:, ['MEDIA ACCELEROMETRO',
               'MINIMO ACCELEROMETRO',
               'MASSIMO ACCELEROMETRO',
               'DEV STD ACCELEROMETRO'
               ]].values
#Allocazione del modello usando il Random Forest
model = ensemble.RandomForestClassifier(n_estimators=100)
#Allenamento del modello con i valori delle features(X) e dei target(y)
model.fit(X, y)
filename = 'model'
#Salvataggio del modello su disco(Model Persistence)
pickle.dump(model, open(filename, 'wb'))
```

Appendice D. Codice sorgente: predictions

```
import pickle
import pandas
model = pickle.load(open("model", 'rb'))
#Lettura del dataset di input di cui non conosciamo gli output
df = pandas.read_csv('input.txt', sep=";")
X = df.loc[:, ['MEDIA ACCELEROMETRO',
               'MINIMO ACCELEROMETRO',
               'MASSIMO ACCELEROMETRO',
               'DEV STD ACCELEROMETRO'
               ]].values
#Salvo le predictions effettuate sui valori di input contenuti in “X”
prediction= model.predict(X)
#Converto il vettore di interi in un vettore di stringhe
prediction= [str(i) for i in prediction]
#Associo il nome del mezzo al corrispettivo numero(Operazione inversa)
for i in range(len(prediction)):
    if prediction[i] == "1":
        prediction[i]="Auto"
    if prediction[i] == "2":
        prediction[i] = "Fermo"
    if prediction[i] == "3":
        prediction[i]="Treno"
    if prediction[i] == "4":
        prediction[i]="A piedi"
    if prediction[i] == "5":
        prediction[i]="Bus"
    if prediction[i] == "6":
        prediction[i]="Bici"
#Aggiunta del campo “PREDICTION” con i suoi valori al df
df['PREDICTION']= prediction
#Scrittura del nuovo df con i valori predetti su un nuovo file
df.to_csv('output.txt', sep=';')
```

Sitografia

Sito ufficiale scikit-learn:

https://scikit-learn.org/stable/

Machine learning:

https://www.slideshare.net/LucaNaso/machine-learning-definizione-e-tipologie

Studio dell'università di Bologna alla quale questo progetto si è ispirato:

http://cs.unibo.it/projects/us-tm2017/index.html