INDICE

1. INTRODUZIONE

L’obiettivo di questo progetto è di sviluppare un sistema di classificazione di genere utilizzando il volto di una persona in modo da determinare se un soggetto è maschio o femmina. Questo obiettivo doveva essere raggiunto tramite il metodo Spider Web che sfrutta i landmark del volto. Per svolgere questo progetto come dataset è stato utilizzato WHE che è composto da 100186 immagini. Una volta applicata la ragnatela è stato utilizzato il classificatore binario SVM per determinare l’accuracy ed altre informazioni utili.

2. PRE-PROCESSING

2.1 Landmark e Ragnatela

Il nostro lavoro è iniziato con l'individuazione dei volti e dei relativi landmark all'interno di ogni immagine del dataset. Per ogni volto presente nell'immagine, vengono individuati 68 landmark. L'algoritmo per individuare i volti ed i relativi landmark è stato utilizzato su un dataset di 100186 immagini e tale elaborazione è andata a buon fine per 76304 immagini. Ciò è dovuto al fatto che all'interno del dataset esistono alcune immagini che non presentano volti, altre immagini che presentano più di un volto senza le relative informazioni per renderli utilizzabili nelle fasi successive ed altre immagini che presentano un unico volto ma in una posa tale da non rendere possibile l'ottenimento dei landmark.

Alla fine di questa prima fase, abbiamo deciso di modificare l'algoritmo fornitoci in modo da ridurre i tempi di elaborazione; per questo abbiamo aggiunto un controllo che scarta immediatamente le immagini in cui è stato riscontrato più di un volto o nessun volto all'interno. Fatto ciò, per le immagini in cui viene riscontrato un unico volto, si prosegue con la costruzione della ragnatela e con l'applicazione della stessa sul volto. Una volta applicata la ragnatela, andiamo a controllare ogni landmark in quale settore della ragnatela ricade; questo perché l'output richiesto da questa fase è il numero di landmark presenti all'interno di ogni settore per ogni immagine.

2.1.1 Configurazioni

Ovviamente, ci è stato richiesto di utilizzare varie configurazioni della ragnatela. Noi abbiamo deciso di utilizzare le migliori sei configurazioni presenti nel paper fornitoci.

Le configurazioni da noi utilizzate sono le seguenti:

1. 4 cerchi e 4 fette per quadrante per un totale di 64 settori. La distanza tra i cerchi adiacenti è di 4/10*raggio, 7/10*raggio e 9/10*raggio.
2. 4 cerchi e 4 fette per quadrante per un totale di 64 settori. La distanza tra i cerchi adiacenti è di 1/15*raggio, 3/15*raggio e 7/15*raggio.
3. 4 cerchi e 4 fette per quadrante per un totale di 64 settori. La distanza tra i cerchi adiacenti è di 8/15*raggio, 12/15*raggio e 14/15*raggio.
4. 4 cerchi e 4 fette per quadrante per un totale di 64 settori. La distanza tra i cerchi adiacenti è di 1/10*raggio, 3/10*raggio e 6/10*raggio.
5. 4 cerchi e 3 fette per quadrante per un totale di 48 settori. La distanza tra i cerchi adiacenti è di raggio/4, raggio/2 e 3/4*raggio.
6. 5 cerchi e 4 fette per quadrante per un totale di 64 settori. La distanza tra i cerchi adiacenti è di raggio/5, 2/5*raggio, 3/5*raggio e 4/5*raggio.

Per fare tutto ciò è stato creato uno script per ogni configurazione della ragnatela con il nome di "calcolo_ragnatelaX.py" dove X sta per il numero della configurazione della

ragnatela. Bisogna notare bene che all'interno dello script è presente anche la parte che si occupa dell'individuazione dei volti e dei relativi landmark, con i relativi controlli per scartare immediatamente le immagini che non presentano unicamente un solo volto, in modo da ridurre i tempi di elaborazione. I tempi di elaborazione di ogni script si attesta sulle 7 ore a configurazione lavorando sul dataset completo composto da poco più di 100.000 immagini.

2.1.2 Risultati dopo l'esecuzione

I risultati dell'elaborazione di ognuno di questi script, vengono salvati all'interno del file "fileX.csv" dove la X sta per il numero della configurazione della ragnatela utilizzata per ottenere tali dati. Ciò ci ha permesso di utilizzare i dati a nostro piacimento in modo immediato, senza dover ogni volta rieseguire gli script precedenti che comunque richiedono tempi abbastanza lunghi.

Bisogna ricordare che questo file contiene esclusivamente il numero di landmark presenti in ogni settore, il numero della foto a cui si riferisce ogni elemento ed il genere associato ad ogni soggetto nella foto.

2.2 Bilanciamento dataset

Alla fine dell'esecuzione delle varie configurazioni della ragnatela, abbiamo controllato che i dati ottenuti fossero bilanciati, per quanto riguarda il genere, attraverso l'apposito script "ControlloGenereX.py" dove la X sta per il numero della configurazione della ragnatela utilizzata per ottenere tali dati. Tramite questo script effettuiamo un controllo sui numeri di maschi e di femmine presenti nei dati ottenuti dalla fase precedente. Grazie all'esecuzione di questo script possiamo notare che il dataset ottenuto è abbastanza sbilanciato, poiché sono presenti solamente 28.687 volti femminili contro i 47.617 volti maschili. È per questo che all'interno dello stesso script abbiamo fatto in modo che venissero scartati un numero sufficiente di dati che permettessero di bilanciare il dataset. Per fare ciò vengono conteggiati il numero di maschi ed il numero di femmine, viene stabilito quale tra questi è minore rispetto all'altro e di conseguenza vengono scartati i dati eccedenti del sesso con il maggior numero di immagini. Eseguito questo script ci ritroveremo con un dataset composto da 57.374 campioni, e con lo stesso numero di soggetti maschili e femminili. Anche in questo caso, per poter lavorare sul nuovo dataset, senza dover ogni volta rieseguire lo script, abbiamo salvato tutti i nuovi dati all'interno del file "fileXfinale.csv" dove la X sta per il numero della configurazione della ragnatela utilizzata per ottenere tali dati.

Bisogna ricordare che anche in questo caso il dataset ottenuto è composto esclusivamente dal numero di landmark presenti in ogni settore, il numero della foto a cui si riferisce ogni elemento ed il genere associato ad ogni soggetto nella foto.

3. CLASSIFICATORE

3.1 Configurazione

Arrivati a questo punto siamo passati alla costruzione del classificatore. Come detto in precedenza, è stato deciso di utilizzare il classificatore binario SVM ed abbiamo suddiviso il dataset finale in 30% di test set e 70% di training set. Abbiamo fatto questo per ogni configurazione ottenuta nella fase precedente andando ad allenare prima il classificatore binario sul training set e successivamente andando a testarlo sul test set. Sui risultati ottenuti abbiamo calcolato l'accuracy totale, la precisione, la recall, f1-score ed il numero di elementi di un genere e dell'altro (support). Inoltre, abbiamo calcolato la macro avg e la weighted avg. Infine, abbiamo deciso di stampare anche la matrice di confusione. Da notare che sono stati costruiti tre classificatori differenti:

1. Il classificatore "Classificatore64settori.py" utilizzato per le configurazioni composte da 64 settori, ovvero le prime quattro configurazioni elencate nei precedenti capitoli;
2. Il classificatore "Classificatore48settori.py" utilizzato per la configurazione composta da 48 settori, ovvero la configurazione numero 5;
3. Il classificatore "Classificatore80settori.py" utilizzato per la configurazione composta da 80 settori, ovvero la configurazione numero 6.

3.2 Risultati ottenuti

Prima di visualizzare i risultati, bisogna tener presente che questi sono stati ottenuti esclusivamente utilizzando il numero di landmark presenti in ogni settore della ragnatela.

Nel caso della prima configurazione della ragnatela abbiamo ottenuto i seguentirisultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.69      0.70      0.70      8502
           1       0.70      0.69      0.70      8711

    accuracy                           0.70     17213
   macro avg       0.70      0.70      0.70     17213
weighted avg       0.70      0.70      0.70     17213


Confusion matrix:
[[5959 2543]
 [2683 6028]]
```

Nel caso della seconda configurazione della ragnatela abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.66      0.68      0.67      8548
           1       0.68      0.66      0.67      8665

    accuracy                           0.67     17213
   macro avg       0.67      0.67      0.67     17213
weighted avg       0.67      0.67      0.67     17213


Confusion matrix:
[[5843 2705]
 [2964 5701]]
```

Nel caso della terza configurazione della ragnatela abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.70      0.68      0.69      8664
           1       0.69      0.70      0.70      8549

    accuracy                           0.69     17213
   macro avg       0.69      0.69      0.69     17213
weighted avg       0.69      0.69      0.69     17213


Confusion matrix:
[[5931 2733]
 [2529 6020]]
```

Nel caso della quarta configurazione della ragnatela abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.71      0.67      0.69      8558
           1       0.69      0.73      0.71      8655

    accuracy                           0.70     17213
   macro avg       0.70      0.70      0.70     17213
weighted avg       0.70      0.70      0.70     17213


Confusion matrix:
[[5703 2855]
 [2326 6329]]
```

Nel caso della quinta configurazione della ragnatela abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.71      0.70      0.71      8577
           1       0.71      0.72      0.71      8636

    accuracy                           0.71     17213
   macro avg       0.71      0.71      0.71     17213
weighted avg       0.71      0.71      0.71     17213


Confusion matrix:
[[5995 2582]
 [2423 6213]]
```

Nel caso della sesta configurazione della ragnatela abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.70      0.67      0.69      8655
           1       0.68      0.71      0.70      8558

    accuracy                           0.69     17213
   macro avg       0.69      0.69      0.69     17213
weighted avg       0.69      0.69      0.69     17213


Confusion matrix:
[[5839 2816]
 [2451 6107]]
```

Come è possibile vedere dalle immagini, i migliori risultati sono stati ottenuti dalla quinta configurazione ottenendo una accuracy del 71% mentre i peggiori risultati sono stati ottenuti dalla seconda configurazione ottenendo una accuracy del 67%. Per quanto riguarda le altre configurazioni, i risultati si equivalgono rimanendo sulla soglia del 69-70%. È possibile visualizzare questi risultati anche nella cartella del progetto; questi prendono il nome di “RisultatiClassificatoreDatiX.PNG”.

3.3 Conclusione

Per concludere, possiamo affermare che con questo metodo non si ottengono risultati eccezionali per quanto riguarda la classificazione del genere ma uno dei pro è sicuramente la velocità del classificatore che può essere allenato e testato in meno di dieci minuti su un campione di circa 57.000 elementi.

4. Fase2

Arrivati a questo punto ci siamo dati l'obiettivo di migliorare il più possibile l'accuracy del classificatore utilizzando anche altri attributi in aggiunta al numero di landmark in ogni settore utilizzati precedentemente. Infatti, all'interno del dataset WHE sono associati ad ogni immagine, oltre al genere, anche l'età, il colore della pelle, la posa, la sorgente intesa come video o immagine, gli occhiali e la dimensione del viso all'interno dell'immagine. Dopo un'attenta analisi, siamo giunti alla conclusione che uno degli attributi che avrebbe potuto far la differenza fosse l'età.

4.1 Pre-Processing

Quindi partendo dai vari dataset "fileXfinale.csv" abbiamo creato, per ognuno di questi, uno script "AggiuntaEtàX.py" dove viene aggiunto l'attributo età ad ogni elemento. Tramite questo script alla fine otterremo un nuovo dataset che abbiamo deciso di chiamare "fileXfinaleConEtà.csv".

4.2 Configurazione

Arrivati a questo punto abbiamo costruito i nuovi classificatori per i nuovi dataset proprio come fatto in precedenza. Ricordiamo che questi classificatori verranno allenati sul numero di landmark presenti in ogni settore e sull'età. Quindi abbiamo costruito i seguenti tre classificatori:

1. Il classificatore "Classificatore64settoriConEtà.py" utilizzato per le configurazioni composte da 64 settori, ovvero le prime quattro configurazioni elencate nei precedenti capitoli, con l'aggiunta dell'età;
2. Il classificatore "Classificatore48settoriConEtà.py" utilizzato per la configurazione composta da 48 settori, ovvero la configurazione numero 5, con l'aggiunta dell'età;
3. Il classificatore "Classificatore80settoriConEtà.py" utilizzato per la configurazione composta da 80 settori, ovvero la configurazione numero 6, con l'aggiuntadell'età.

4.3 Risultati ottenuti

Nel caso della prima configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.73      0.74      0.73      8629
           1       0.73      0.72      0.73      8584

    accuracy                           0.73     17213
   macro avg       0.73      0.73      0.73     17213
weighted avg       0.73      0.73      0.73     17213


Confusion matrix:
[[6371 2258]
 [2374 6210]]
```

Nel caso della seconda configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.70      0.76      0.73      8600
           1       0.74      0.68      0.71      8613

    accuracy                           0.72     17213
   macro avg       0.72      0.72      0.72     17213
weighted avg       0.72      0.72      0.72     17213


Confusion matrix:
[[6519 2081]
 [2771 5842]]
```

Nel caso della terza configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.73      0.75      0.74      8592
           1       0.74      0.72      0.73      8621

    accuracy                           0.73     17213
   macro avg       0.73      0.73      0.73     17213
weighted avg       0.73      0.73      0.73     17213


Confusion matrix:
[[6403 2189]
 [2424 6197]]
```

Nel caso della quarta configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.73      0.73      0.73      8654
           1       0.73      0.73      0.73      8559

    accuracy                           0.73     17213
   macro avg       0.73      0.73      0.73     17213
weighted avg       0.73      0.73      0.73     17213


Confusion matrix:
[[6306 2348]
 [2324 6235]]
```

Nel caso della quinta configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.74      0.74      0.74      8656
           1       0.73      0.73      0.73      8557

    accuracy                           0.73     17213
   macro avg       0.73      0.73      0.73     17213
weighted avg       0.73      0.73      0.73     17213


Confusion matrix:
[[6391 2265]
 [2302 6255]]
```

Nel caso della sesta configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.74      0.73      0.74      8693
           1       0.73      0.73      0.73      8520

    accuracy                           0.73     17213
   macro avg       0.73      0.73      0.73     17213
weighted avg       0.73      0.73      0.73     17213


Confusion matrix:
[[6386 2307]
 [2287 6233]]
```

Come è possibile notare dai risultati ottenuti, aggiungendo l'età otteniamo un buon miglioramento dei risultati rispetto all'utilizzo esclusivo del numero di landmark all'interno dei settori. Utilizzando l'età tutte le configurazioni raggiungono un'accuracy del 73% ad eccezione della configurazione numero 2 che ottiene il 72% dell'accuracy. Possiamo notare che i minori miglioramenti sono ottenuti dalla configurazione numero 5 che in precedenza era la migliore (questa riesce comunque a passare dal 70% di accuracy al 73%) mentre i maggiori miglioramenti sono ottenuti dalla configurazione numero 2 (questa riesce a passare dal 67% di accuracy al 72%). Nonostante ciò la configurazione numero 2 rimane quella con la peggior accuracy.

È possibile visualizzare questi risultati anche nella cartella del progetto; questi prendono il nome di "RisultatiClassificatoreDatiConEtàX.PNG".

4.4 Conclusione

Possiamo concludere che è conveniente utilizzare anche l'età per allenare il classificatore poiché in termini di prestazioni il classificatore non ha peggiorato i tempi di elaborazione che si attestano sempre sotto i dieci minuti. Inoltre, i risultati della classificazione del genere vengono influenzati in positivo.

5. Fase3

Arrivati a questo punto ci siamo dati ancora una volta l'obiettivo di migliorare il più possibile l'accuracy del classificatore utilizzando ulteriori attributi in aggiunta al numero di landmark in ogni settore ed all'età utilizzati precedentemente. Dopo un'attenta analisi, siamo giunti alla conclusione che altri attributi che avrebbe potuto far la differenza fossero la posa, il colore della pelle e la dimensione del viso all'interno dell'immagine.

5.1 Pre-Processing

Quindi partendo dai vari dataset "fileXfinaleConEtà.csv" abbiamo creato, per ognuno di questi, uno script "AggiuntaPosaColoreDimensioneVisoX.py" dove vengono aggiunti gli attributi posa, colore e dimensione del viso all'interno dell'immagine ad ogni elemento. Tramite questo script alla fine otterremo un nuovo dataset che abbiamo deciso di chiamare "fileXfinaleConEtàPosaColoreDimensioneViso.csv".

5.2 Configurazione

Arrivati a questo punto abbiamo costruito i nuovi classificatori per i nuovi dataset proprio come fatto in precedenza. Ricordiamo che questi classificatori verranno allenati sul numero

di landmark presenti in ogni settore, sull'età, sulla posa, sul colore del viso e sulla dimensione del viso. Quindi abbiamo costruito i seguenti tre classificatori:

4. Il classificatore "Classificatore64settoriConEtàPosaColoreDimensioneViso.py" utilizzato per le configurazioni composte da 64 settori, ovvero le prime quattro configurazioni elencate nei precedenti capitoli, con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso;
5. Il classificatore "Classificatore48settoriConEtàPosaColoreDimensioneViso.py" utilizzato per la configurazione composta da 48 settori, ovvero la configurazione numero 5, con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso;
6. Il classificatore "Classificatore80settoriConEtàPosaColoreDimensioneViso.py" utilizzato per la configurazione composta da 80 settori, ovvero la configurazione numero 6, con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso.

5.3 Risultati ottenuti

Nel caso della prima configurazione della ragnatela con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.73      0.76      0.74      8570
           1       0.75      0.73      0.74      8643

    accuracy                           0.74     17213
   macro avg       0.74      0.74      0.74     17213
weighted avg       0.74      0.74      0.74     17213


Confusion matrix:
[[6494 2076]
 [2374 6269]]
```

Nel caso della seconda configurazione della ragnatela con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.69      0.75      0.72      8592
           1       0.73      0.67      0.70      8621

    accuracy                           0.71     17213
   macro avg       0.71      0.71      0.71     17213
weighted avg       0.71      0.71      0.71     17213


Confusion matrix:
[[6460 2132]
 [2840 5781]]
```

Nel caso della terza configurazione della ragnatela con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.72      0.75      0.74      8562
           1       0.74      0.72      0.73      8651

    accuracy                           0.73     17213
   macro avg       0.73      0.73      0.73     17213
weighted avg       0.73      0.73      0.73     17213


Confusion matrix:
[[6432 2130]
 [2442 6209]]
```

Nel caso della quarta configurazione della ragnatela con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.73      0.74      0.73      8527
           1       0.74      0.73      0.73      8686

    accuracy                           0.73     17213
   macro avg       0.73      0.73      0.73     17213
weighted avg       0.73      0.73      0.73     17213


Confusion matrix:
[[6281 2246]
 [2366 6320]]
```

Nel caso della quinta configurazione della ragnatela con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.74      0.75      0.74      8740
           1       0.74      0.73      0.73      8473

    accuracy                           0.74     17213
   macro avg       0.74      0.74      0.74     17213
weighted avg       0.74      0.74      0.74     17213


Confusion matrix:
[[6525 2215]
 [2279 6194]]
```

Nel caso della sesta configurazione della ragnatela con l'aggiunta dell'età, della posa, del colore del viso e della dimensione del viso abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.73      0.75      0.74      8628
           1       0.74      0.72      0.73      8585

    accuracy                           0.74     17213
   macro avg       0.74      0.74      0.74     17213
weighted avg       0.74      0.74      0.74     17213


Confusion matrix:
[[6446 2182]
 [2377 6208]]
```

Come è possibile notare dai risultati ottenuti, aggiungendo l'età, la posa, il colore del viso e la dimensione del viso otteniamo un risultato non molto differente rispetto all'utilizzo del numero di landmark all'interno dei settori e all'utilizzo dell'età. Utilizzando l'età, la posa, il colore del viso e la dimensione del viso tutte le configurazioni raggiungono un'accuracy del 74% ad eccezione delle configurazioni numero 3 e 4 che restano fisse a 73% dell'accuracy. Invece la configurazione numero 2, che con solo l'età era di 72%, scende ad una accuracy del 71%. Possiamo notare che la configurazione numero 2 rimane quella con la peggior accuracy.

È possibile visualizzare questi risultati anche nella cartella del progetto; questi prendono il nome di "RisultatiClassificatoreDatiConEtàPosaColoreDimensioneVisoX.PNG".

5.4 Conclusione

Possiamo concludere che non è molto conveniente utilizzare anche la posa, il colore del viso e la dimensione del viso per allenare il classificatore poiché in termini di prestazioni il

classificatore ha peggiorato i tempi di elaborazione che si attestano sui quindici/venti minuti. Inoltre, i risultati della classificazione del genere vengono influenzati di poco in positivo.

6. Conclusioni

Nella seguente immagine viene riportata una tabella riassuntiva dei risultati ottenuti nelle 3 Fasi.

Configurazione	Accuracy senza età	Accuracy con età	Accuracy con tutto
1	0,70	0,73	0,74
2	0,67	0,72	0,71
3	0,69	0,73	0,73
4	0,70	0,73	0,73
5	0,71	0,73	0,74
6	0,69	0,73	0,74

Come è possibile vedere l'accuracy migliore si ottiene utilizzando i landmark del volto, l'età, la posa, il colore del viso e la dimensione del viso; ciò però comporta ad un incremento dei tempi di elaborazione che a nostro avviso non giustificano tale utilizzo. È molto più conveniente utilizzare solamente i landmark del volto e l'età poiché i tempi di elaborazione sono praticamente gli stessi rispetto all'utilizzo dei soli landmark del volto ed inoltre si ottiene un buon incremento per quanto riguarda l'accuracy.

Arrivati a questo punto possiamo dire che gli obiettivi a noi assegnati sono stati raggiunti e che il metodo Spider Web sia un'opzione interessante e valida per sviluppare un sistema di classificazione di genere. In termini di tempi di elaborazione del classificatore sono molto soddisfacenti e anche il tempo di elaborazione del pre-processing sono più che accettabili per il quantitativo di immagini elaborate. In termini di accuracy utilizzando il dataset WHE ed il classificatore binario SVM i risultati sono buoni (74% di accuracy) in rapporto alle tempistiche, ma non sono comparabili ad altri metodi conosciuti in letteratura i quali raggiungono circa 95% di accuracy con lo svantaggio delle tempistiche.

7. APPROFONDIMENTO SOMMESE

Arrivati a questo punto ci è stato chiesto di provare ad esaminare i risultati ottenuti utilizzando il numero di landmark presenti in ogni settore del volto e la dimensione del viso all'interno dell'immagine. Questo per capire appunto quanto la dimensione del viso all'interno dell'immagine possa influire per quanto riguarda la classificazione di genere.

7.1 Pre-Processing

Quindi partendo dai vari dataset "fileXfinale.csv" abbiamo creato, per ognuno di questi, uno script "AggiuntaDimensioneVisoX.py" dove viene aggiunto l'attributo dimensione del viso ad ogni elemento. Tramite questo script alla fine otterremo un nuovo dataset che abbiamo deciso di chiamare "fileXfinaleConDimensioneVolto.csv".

7.2 Configurazione

Arrivati a questo punto abbiamo costruito i nuovi classificatori per i nuovi dataset proprio come fatto in precedenza. Ricordiamo che questi classificatori verranno allenati sul numero di landmark presenti in ogni settore e sulla dimensione del volto. Quindi abbiamo costruito i seguenti tre classificatori:

1. Il classificatore "Classificatore64settoriConEtà.py" utilizzato per le configurazioni composte da 64 settori, ovvero le prime quattro configurazioni elencate nei precedenti capitoli, con l'aggiunta della dimensione del volto;
2. Il classificatore "Classificatore48settoriConEtà.py" utilizzato per la configurazione composta da 48 settori, ovvero la configurazione numero 5, con l'aggiunta della dimensione del volto;
3. Il classificatore "Classificatore80settoriConEtà.py" utilizzato per la configurazione composta da 80 settori, ovvero la configurazione numero 6, con l'aggiuntadella dimensione del volto.

7.3 Risultati ottenuti

Nel caso della prima configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.70      0.70      0.70      8593
           1       0.70      0.70      0.70      8620

    accuracy                           0.70     17213
   macro avg       0.70      0.70      0.70     17213
weighted avg       0.70      0.70      0.70     17213


Confusion matrix:
[[6016 2577]
 [2627 5993]]
```

Nel caso della seconda configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.67      0.69      0.68      8601
           1       0.68      0.67      0.67      8612

    accuracy                           0.68     17213
   macro avg       0.68      0.68      0.68     17213
weighted avg       0.68      0.68      0.68     17213


Confusion matrix:
[[5919 2682]
 [2863 5749]]
```

Nel caso della terza configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.70      0.70      0.70      8629
           1       0.70      0.70      0.70      8584

    accuracy                           0.70     17213
   macro avg       0.70      0.70      0.70     17213
weighted avg       0.70      0.70      0.70     17213


Confusion matrix:
[[6072 2557]
 [2617 5967]]
```

Nel caso della quarta configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.72      0.67      0.69      8616
           1       0.69      0.73      0.71      8597

    accuracy                           0.70     17213
   macro avg       0.70      0.70      0.70     17213
weighted avg       0.70      0.70      0.70     17213


Confusion matrix:
[[5775 2841]
 [2281 6316]]
```

Nel caso della quinta configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.71      0.70      0.70      8556
           1       0.71      0.72      0.71      8657

    accuracy                           0.71     17213
   macro avg       0.71      0.71      0.71     17213
weighted avg       0.71      0.71      0.71     17213


Confusion matrix:
[[5971 2585]
 [2467 6190]]
```

Nel caso della sesta configurazione della ragnatela con l'aggiunta dell'età abbiamo ottenuto i seguenti risultati:

```
Classification report for classifier SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       0.70      0.69      0.69      8593
           1       0.69      0.71      0.70      8620

    accuracy                           0.70     17213
   macro avg       0.70      0.70      0.70     17213
weighted avg       0.70      0.70      0.70     17213


Confusion matrix:
[[5912 2681]
 [2533 6087]]
```

Come è possibile notare dai risultati ottenuti, aggiungendo la dimensione del volto all'interno dell'immagine, otteniamo un minimo miglioramento dei risultati rispetto all'utilizzo esclusivo del numero di landmark all'interno dei settori. Infatti, la configurazione numero 2 passa dal 67% al 68% di accuracy e le configurazioni numero 3 e 6 passano dal 69% al 70% di accuracy; le rimanenti configurazioni rimangono inalterati i precedenti risultati. La configurazione migliore rimane sempre la configurazione numero 5 che però non ottiene miglioramenti con questa aggiunta. Per quanto riguarda la configurazione peggiore, questa rimane la configurazione numero 2 che però riesce ad ottenere un miglioramento dell'1% utilizzando anche la dimensione del volto.

È possibile visualizzare questi risultati anche nella cartella del progetto; questi prendono il nome di "RisultatiClassificatoreDatiConDimensioneVisoX.PNG".

7.4 Conclusione

Possiamo concludere che potrebbe essere conveniente utilizzare anche la dimensione del volto per allenare il classificatore poiché in termini di prestazioni il classificatore non ha peggiorato i tempi di elaborazione che si attestano sempre sotto i dieci minuti. Però, bisogna pur sempre ricordare che questa aggiunta migliora l'accuracy solamente in tre configurazioni e che lo fa con un minimo miglioramento dell'1%.

8. APPROFONDIMENTO CESARANO

Arrivati a questo punto ci è stato chiesto di provare ad esaminare i risultati ottenuti utilizzando il classificatore KNN sul numero di landmark presenti in ogni settore del volto. Questo andava applicato soltanto per la configurazione in cui abbiamo ottenuto la migliore accuracy, ovvero la 5(4 cerchi e 3 fette per quadrante), la cui accuracy è stata del 71%. Questo per capire se il classificatore KNN ci permette di avere un accuracy migliore.

8.1 KNN

Il K-Nearest Neighbors (KNN) è uno degli algoritmi più conosciuti nel machine learning che, oltre alla sua semplicità, produce buoni risultati in un gran numero di domini. KNN è un

algoritmo di apprendimento supervisionato, il cui scopo è quello di predire una nuova istanza conoscendo i data points che sono separati in diverse classi.
L'algoritmo funziona brevemente in questo modo:
Supponiamo che il nostro dataset di training sia composto da due classi di esempi: classe A e classe B, come nel nostro caso (A=maschio, B=femmine). Se al modello applichiamo un nuovo ingresso da classificare, questo verrà portato nello spazio delle feature e verrà confrontato con tutti gli altri esempi del dataset fino a trovare l'esempio più vicino. La classe del nuovo ingresso sarà la stessa dell'esempio trovato. Se piuttosto di cercare il più vicino ne cerca diversi(k), la classe di appartenenza dell'ingresso sarà quella che è maggiormente rappresentata tra gli esempi trovati. Es. se k=3 ed i 3 vicini sono 2 maschi e 1 femmina allora la classe del nuovo dato in ingresso sarà la classe maschio.

8.2 Configurazione

Come appena detto, abbiamo dovuto utilizzare il classificatore binario KNN e nello stesso modo per quando abbiamo utilizzato il classificatore SVM, abbiamo suddiviso il dataset finale in 30% di test set e 70% di training set. Sui risultati ottenuti abbiamo calcolato l'accuracy totale, la precisione, la recall, f1-score ed il numero di elementi di un genere e dell'altro (support). Inoltre, abbiamo calcolato la macro avg e la weighted avg. Infine, abbiamo deciso di stampare anche la matrice di confusione.
Quindi partendo dal dataset "file5finale.csv" abbiamo costruito il seguente classificatore:

1. Il classificatore "Classificatore48settoriKNN.py" utilizzato per la configurazione composta da 48 settori, ovvero la configurazione numero 5(4 cerchi e 3 fette per quadrante);

8.3 Risultati ottenuti

Abbiamo testato questo classificatore con varie istanze di k ma ne riportiamo soltanto 3, quando k=1, quando k=5 (quello di default) e quando k=13. Nel caso in cui abbiamo scelto k=1, abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier KNeighborsClassifier(n_neighbors=1):
              precision    recall  f1-score   support

           0       0.67      0.67      0.67      8649
           1       0.67      0.67      0.67      8564

    accuracy                           0.67     17213
   macro avg       0.67      0.67      0.67     17213
weighted avg       0.67      0.67      0.67     17213


Confusion matrix:
[[5789 2860]
 [2854 5710]]
```

Nel caso in cui abbiamo scelto k=5, abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier KNeighborsClassifier():
              precision    recall  f1-score   support

           0       0.71      0.69      0.70      8648
           1       0.70      0.71      0.70      8565

    accuracy                           0.70     17213
   macro avg       0.70      0.70      0.70     17213
weighted avg       0.70      0.70      0.70     17213


Confusion matrix:
[[6003 2645]
 [2494 6071]]
```

Nel caso in cui abbiamo scelto k=13, abbiamo ottenuto i seguenti risultati:

```
Python 3.7.7 (default, Mar 23 2020, 23:19:08) [MSC v.1916 64 bit (AMD64)]
Classification report for classifier KNeighborsClassifier(n_neighbors=13):
              precision    recall  f1-score   support

           0       0.71      0.70      0.71      8564
           1       0.71      0.72      0.71      8649

    accuracy                           0.71     17213
   macro avg       0.71      0.71      0.71     17213
weighted avg       0.71      0.71      0.71     17213
```

Abbiamo scelto k=5 perché era il valore consigliato, invece abbiamo scelto k=1 e k=13 perché sono stati rispettivamente i risultati con la peggiore e la migliore accuracy.

Come è possibile notare dai risultati ottenuti, utilizzando il classificatore KNN, otteniamo un accuracy del 67% quando k=1, un accuracy del 70% quando k=5 ed un accuracy del 71% quando k=13. La configurazione numero 5, utilizzando il classificatore SVM con l'utilizzo esclusivo del numero di landmark all'interno dei settori, è di 71%. Detto ciò si può notare che invece con il classificatore KNN otteniamo un peggioramento di accuracy quando viene utilizzato k=1 e k=5 ed un valore stabile di accuracy quando viene utilizzato k=13.

È possibile visualizzare questi risultati anche nella cartella del progetto; questi prendono il nome di "RisultatiClassificatoreKNNvicini=Y.PNG", dove Y rappresenta il valore di k preso in considerazione.

8.4 Conclusione

Possiamo concludere che potrebbe essere conveniente utilizzare il classificatore KNN poiché in termini di prestazioni il classificatore non ha peggiorato i tempi di elaborazione che si attestano sempre sotto i dieci minuti. Però, bisogna pur sempre ricordare che non c'è stata nessuna miglioria nella configurazione numero 5 con nessun valore di k.