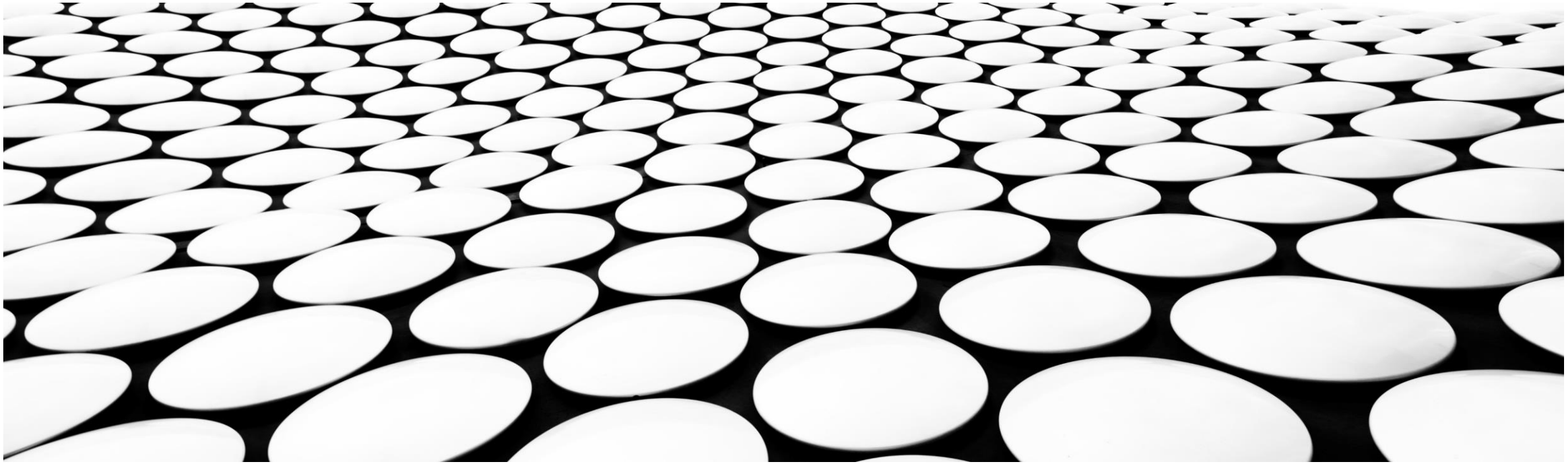
PROGETTO MACHINE LEARNING A.A. 2020/2021

EMILIO CASELLA MATR.204898



CARICAMENTO DEL DATASET IMMAGINI

- Il dataset comprende quattro tipologie di immagini da analizzare: jeans,shirts,trousers,watches.
  Le immagini vengono caricate in scala di grigio, normalizzate e "flattenizzate" in un array numpy, in modo da agevolare le operazioni successive.Il dataset è formato da (#esempi x (80 x 60), target).

TUNING DEI MODELLI SVC, ADABOOST E KNN

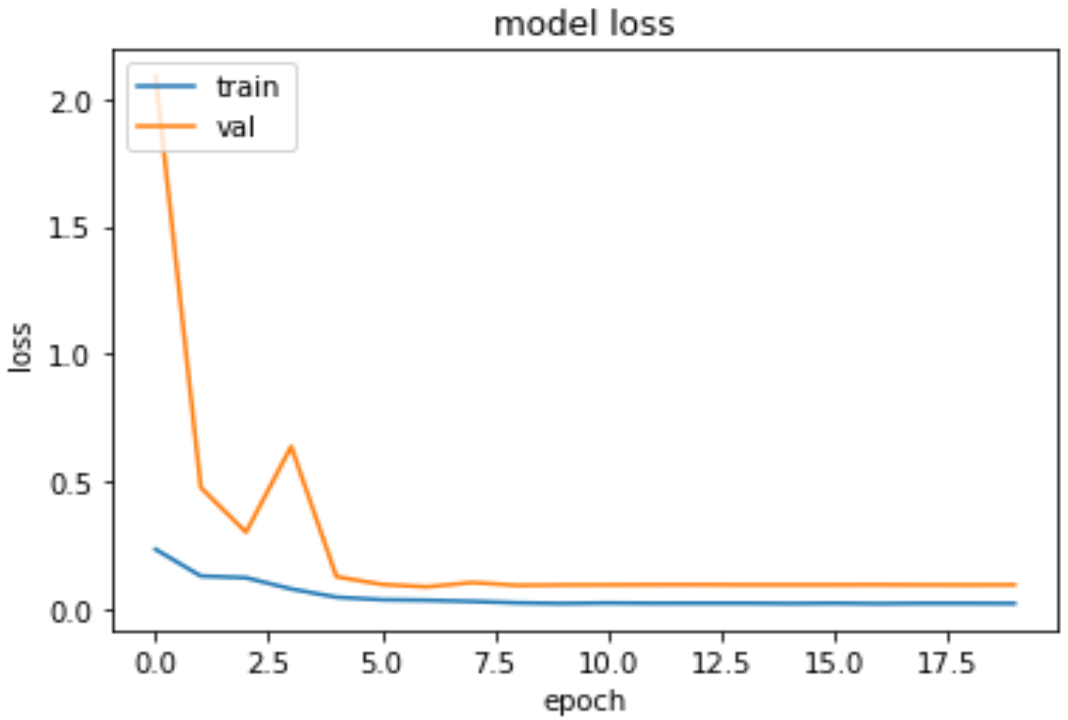
```
MyClassificatorTrain().transform(X,y)
```

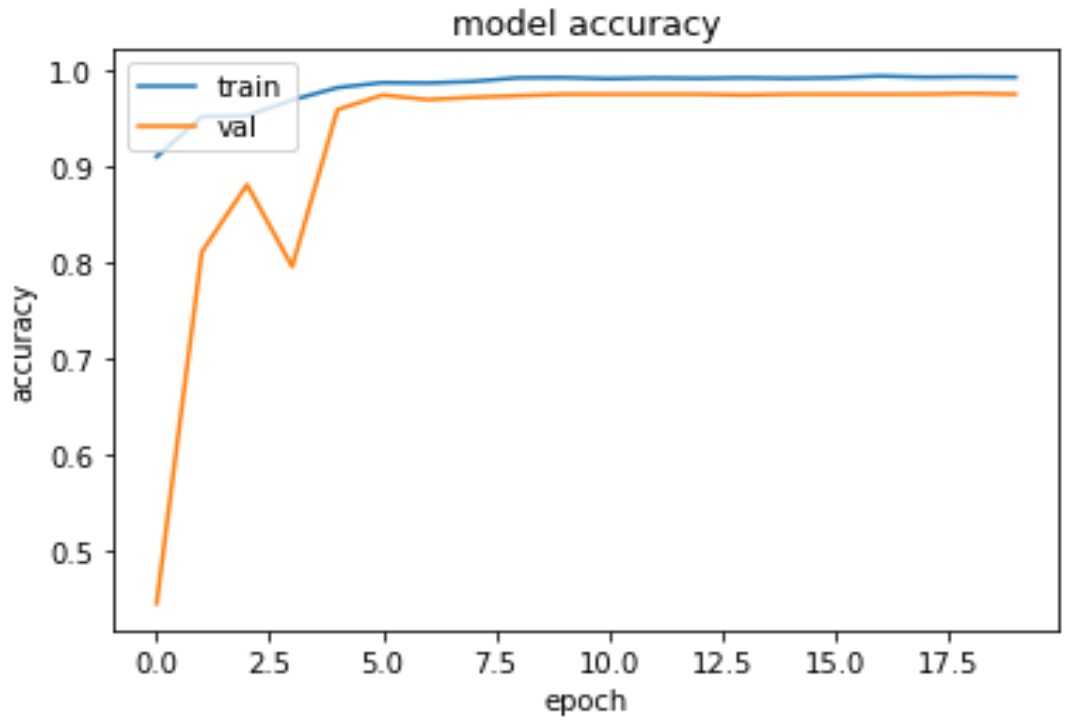
```
Training modelli...
AdaBoost...
Best Param: {'learning_rate': 1.0, 'n_estimators': 500}
Tuning SVC...
Best Param: {'C': 150, 'gamma': 'scale', 'kernel': 'rbf'}
Tuning KNN...
Best Param: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Fatto!
```

RETE NEURALE

```
def create_network():
    opt = keras.optimizers.Adam(learning_rate=0.001)
    model_conv = keras.Sequential()
    model_conv.add(layers.Conv2D(25, (5, 5), activation='relu', input_shape=(i, j, 1)))
    model_conv.add(layers.MaxPooling2D((2, 2)))
    model_conv.add(layers.Conv2D(50, (5, 5), activation='relu'))
    model_conv.add(layers.MaxPooling2D((2, 2)))
    model_conv.add(layers.BatchNormalization())
    model_conv.add(layers.Conv2D(70, (3, 3), activation='relu'))
    model_conv.add(layers.MaxPooling2D((2, 2)))
    model_conv.add(layers.BatchNormalization())
    model_conv.add(layers.Flatten())
    model_conv.add(Dense(units=100, activation='relu'))
    model_conv.add(Dense(units=100, activation='relu'))
    model_conv.add(Dropout(0.25))
    model_conv.add(layers.Dense(4, activation='softmax'))
    model_conv.compile(loss='sparse_categorical_crossentropy', optimizer=opt, metrics=['accuracy']);
    return model_conv
```

RETE NEURALE



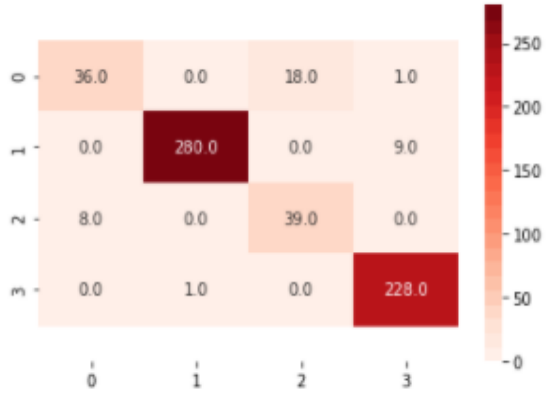


ACCURATEZZA SULLA 10-FOLD CROSS-VALIDATION

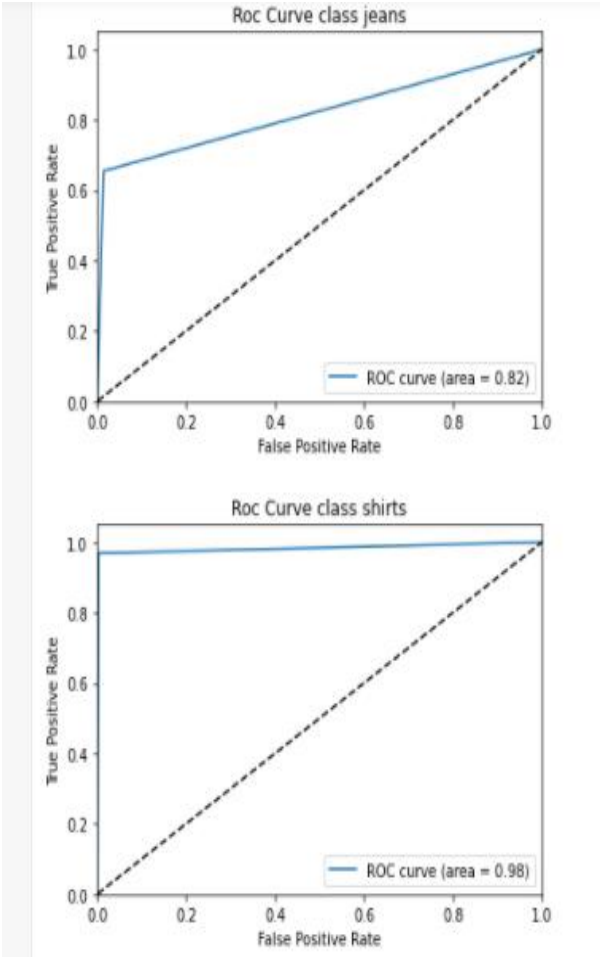
- 1)Precision: La precisione è l'abilità di un classificatore di non etichettare un'istanza positiva che è in realtà negativa. Per ogni classe è definito come il rapporto tra veri positivi e la somma di veri e falsi positivi;
- 2)Recall: detta anche sensitivity o true positive, è la capacità di un classificatore di trovare tutte le istanze positive. Per ogni classe è definito come il rapporto tra i veri positivi e la somma dei veri positivi e dei falsi negativi;
- 3)F-score: è una media armonica ponderata delle metriche Precision e Recall in modo tale che il punteggio migliore sia 1 e il peggiore sia 0;
- 4)Matrice di confusione: permette di visualizzare meglio il rapporto tra TP,FP,TN,FN;
- 5)Curve di ROC: calcola la proporzione di veri positivi (la sensibilità) e la proporzione di falsi positivi.

ADABOOST

```
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.9178744  0.8921095  0.9178744  0.91465378 0.91935484 0.93064516
 0.94032258 0.92258065 0.91774194 0.92741935]
Ottenendo un'accuratezza media di 0.92 con deviazione standard di 0.012
La migliore accuratezza e' 0.94 ottenuta nello split numero 7
Salvo il modello ottenuto nello split numero 7
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 7
```

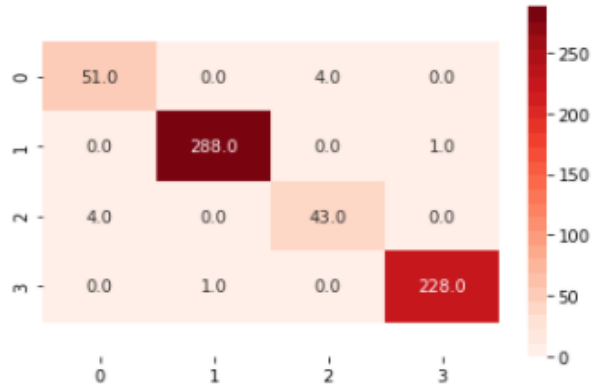


	precision	recall	f1-score	support
jeans	0.82	0.65	0.73	55
shirts	1.00	0.97	0.98	289
trousers	0.68	0.83	0.75	47
watches	0.96	1.00	0.98	229
accuracy			0.94	620
macro avg	0.86	0.86	0.86	620
weighted avg	0.94	0.94	0.94	620

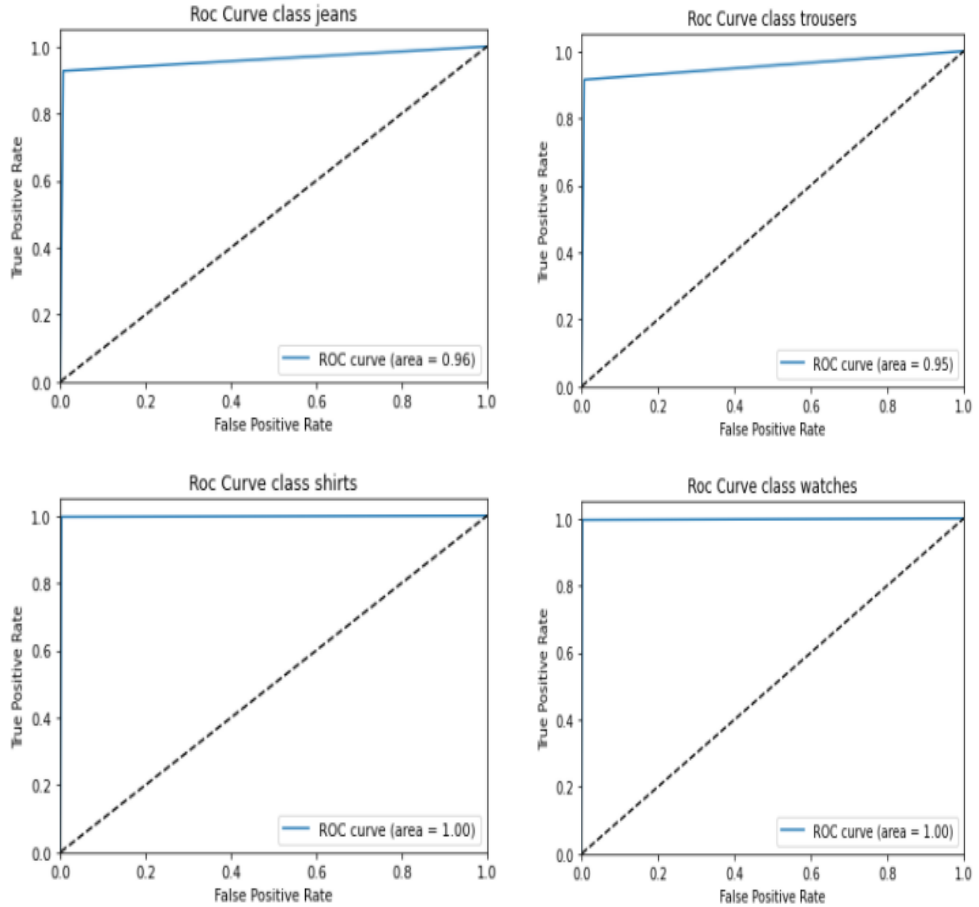


SVC

```
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.97101449 0.97584541 0.97584541 0.97584541 0.98387097 0.98064516
 0.98225806 0.97741935 0.97258065 0.97419355]
Ottenendo un'accuratezza media di 0.977 con deviazione standard di 0.004
La migliore accuratezza e' 0.984 ottenuta nello split numero 5
Salvo il modello ottenuto nello split numero 5
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 5
```



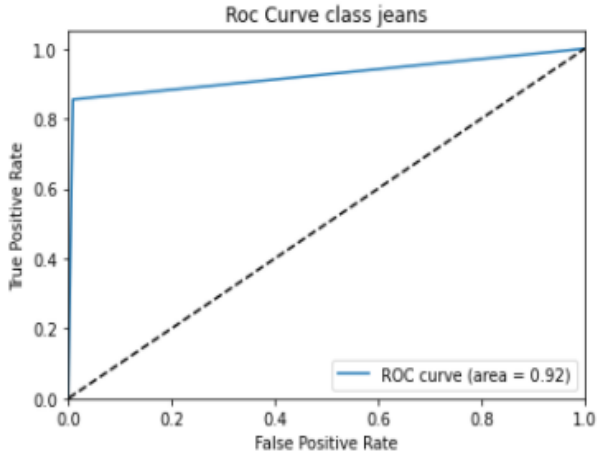
	precision	recall	f1-score	support
jeans	0.93	0.93	0.93	55
shirts	1.00	1.00	1.00	289
trousers	0.91	0.91	0.91	47
watches	1.00	1.00	1.00	229
accuracy			0.98	620
macro avg	0.96	0.96	0.96	620
weighted avg	0.98	0.98	0.98	620

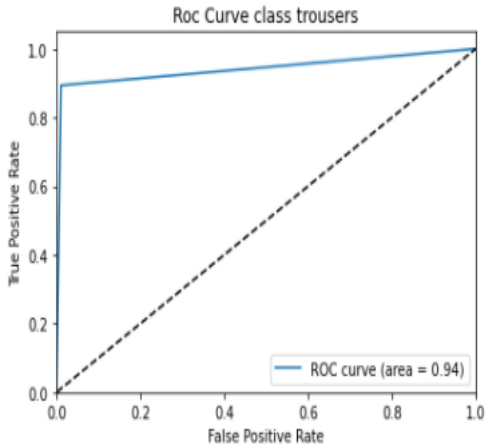


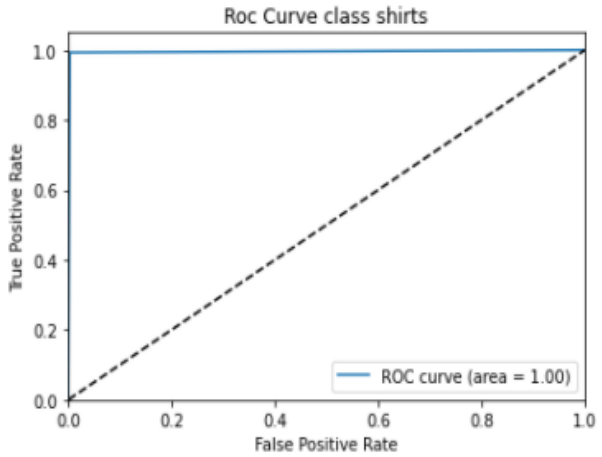
KNN

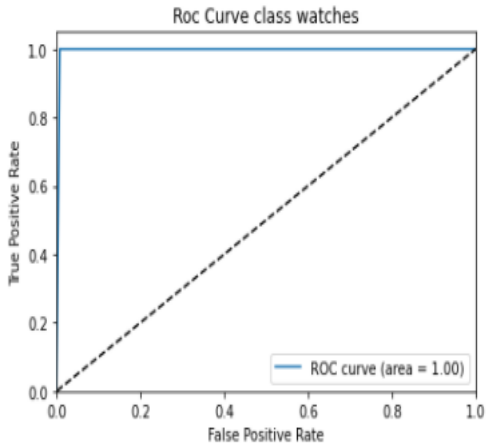
```
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.97101449 0.96940419 0.96618357 0.96940419 0.97580645 0.95483871
 0.97096774 0.96451613 0.94677419 0.96451613]
Ottenendo un'accuratezza media di 0.965 con deviazione standard di 0.008
La migliore accuratezza e' 0.976 ottenuta nello split numero 5
Salvo il modello ottenuto nello split numero 5
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 5
```

	precision	recall	f1-score	support
jeans	0.90	0.85	0.88	55
shirts	1.00	0.99	0.99	289
trousers	0.88	0.89	0.88	47
watches	0.99	1.00	0.99	229
accuracy			0.98	620
macro avg	0.94	0.94	0.94	620
weighted avg	0.98	0.98	0.98	620



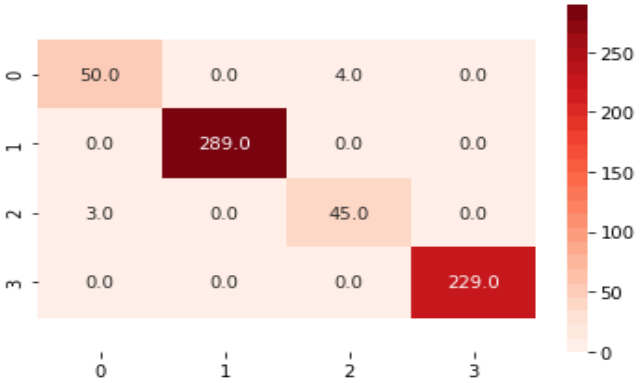




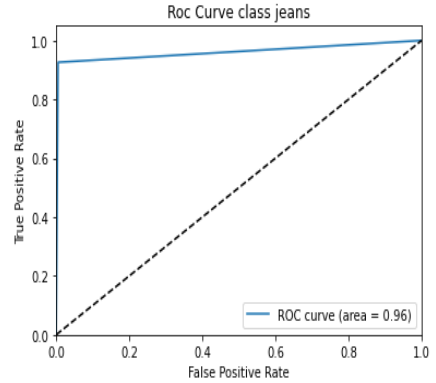


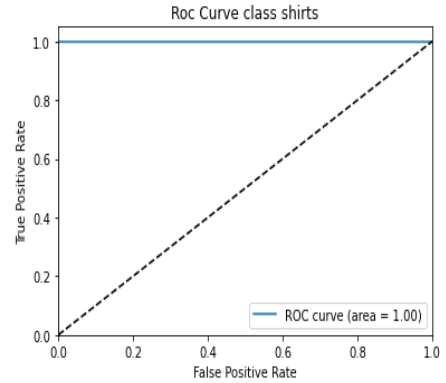
NN

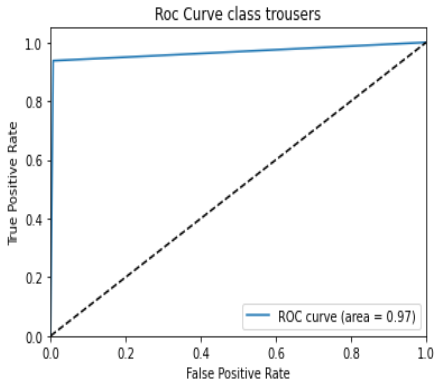
```
Score per fold
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.9758453965187073, 0.9838969111442566, 0.977455735206604, 0.9710144996643066, 0.982258081436157
2, 0.9822580814361572, 0.9854838848114014, 0.9887096881866455, 0.9661290049552917, 0.9741935729980
469]
Ottenendo un'accuratezza media di 0.979 con deviazione standard di 0.007
La migliore accuratezza e' 0.989 ottenuta nello split numero 8
Salvo il modello ottenuto nello split numero 8
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 8
```

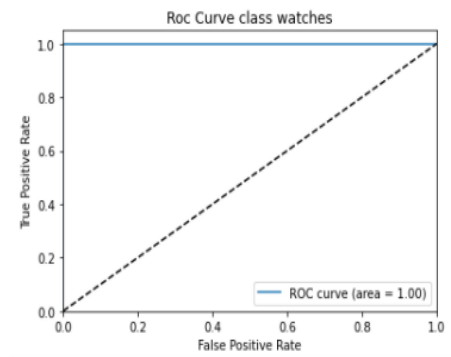


	precision	recall	f1-score	support
0	0.94	0.93	0.93	54
1	1.00	1.00	1.00	289
2	0.92	0.94	0.93	48
3	1.00	1.00	1.00	229
accuracy			0.99	620
macro avg	0.97	0.97	0.97	620
weighted avg	0.99	0.99	0.99	620



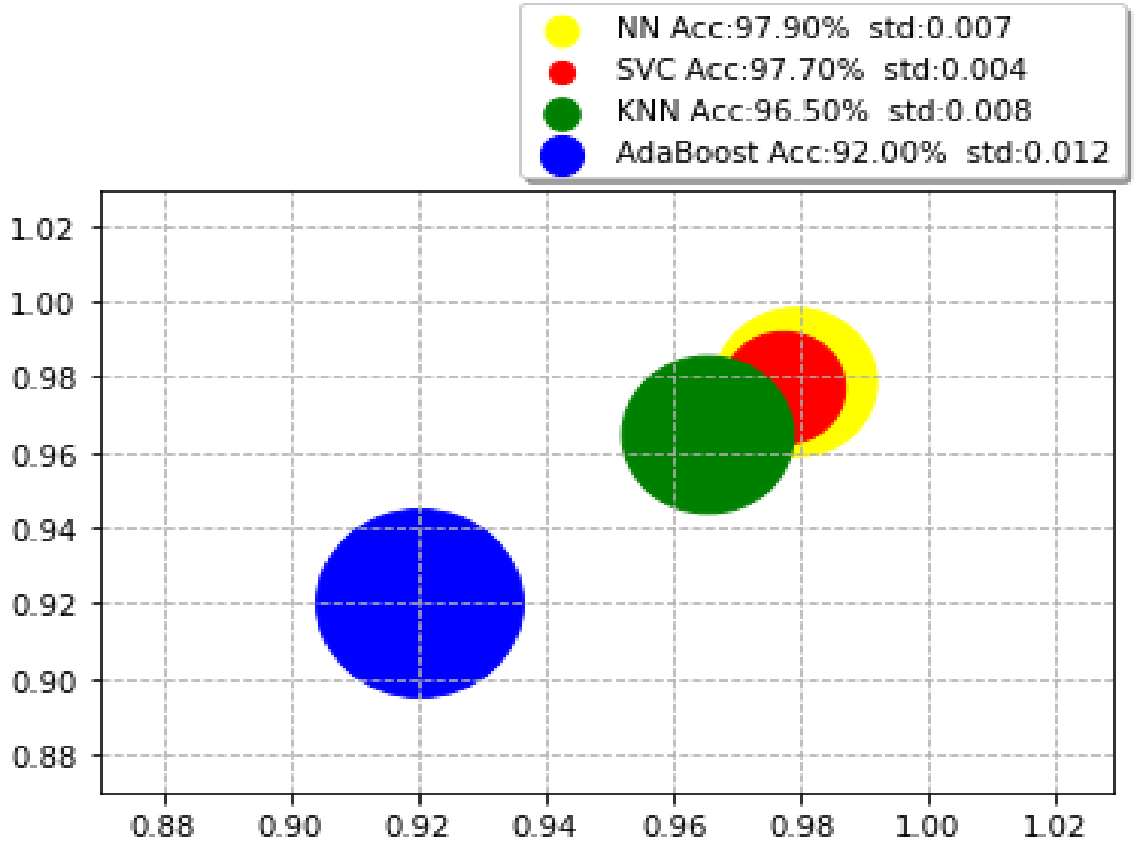




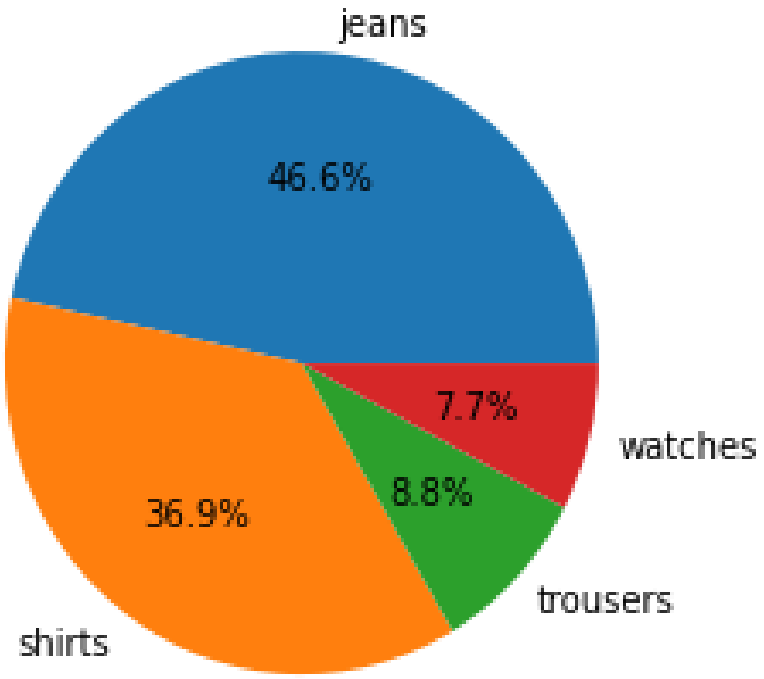


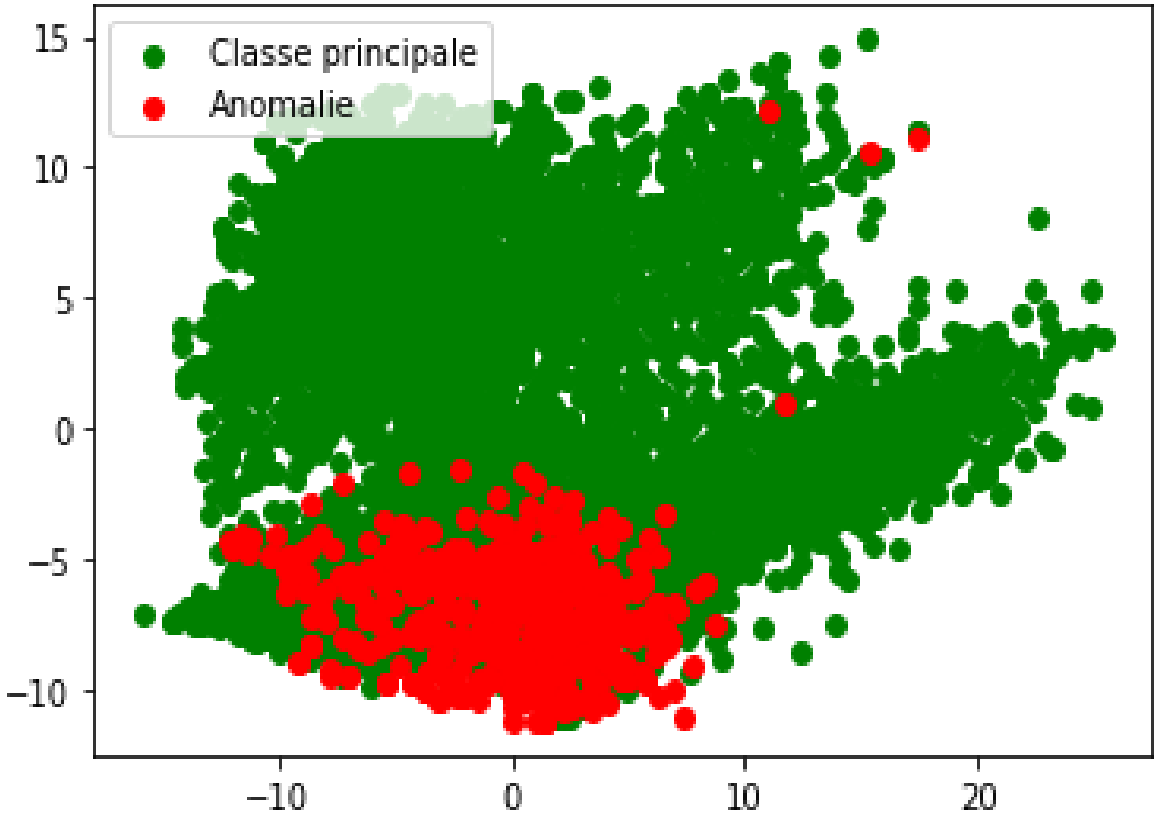
ACCURATEZZA SULLA 10-FOLD CROSS-VALIDATION

	accuracy_mean	std
NN	0.979	0.007
SVC	0.977	0.004
KNN	0.965	0.008
AdaBoost	0.920	0.012



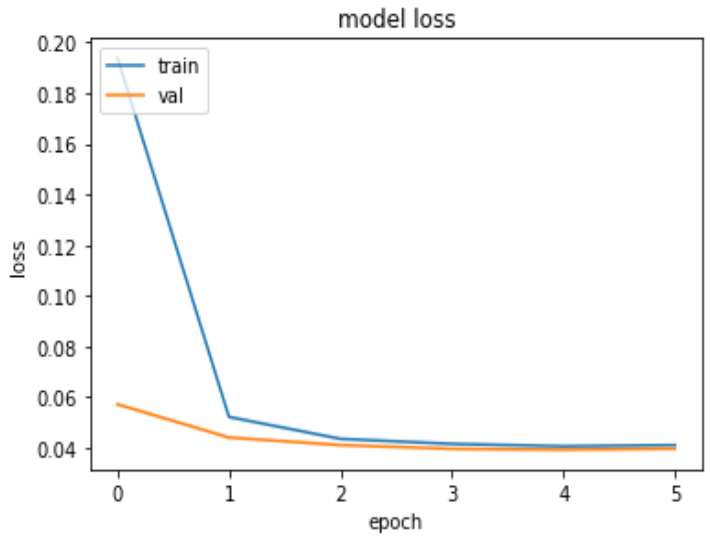
ANOMALY DETECTION

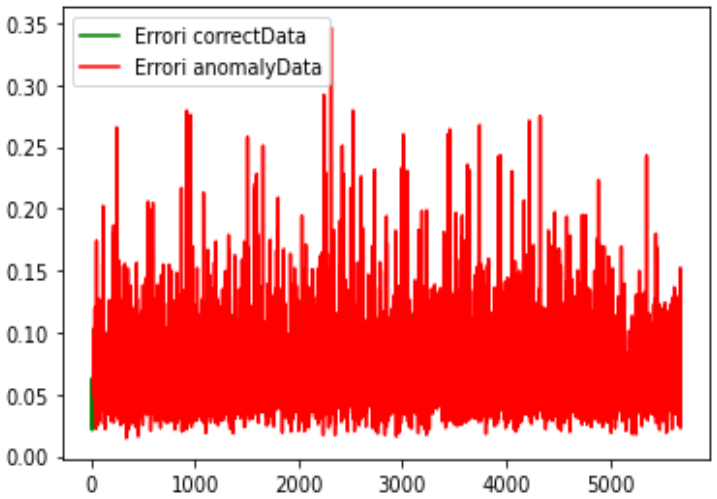


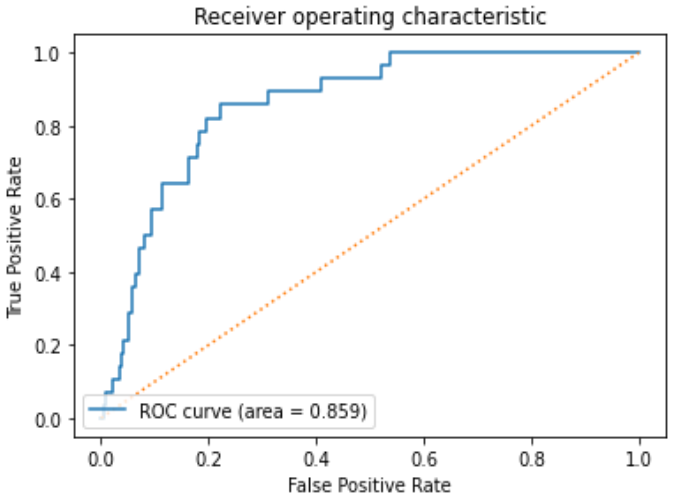


RETE NEURALE

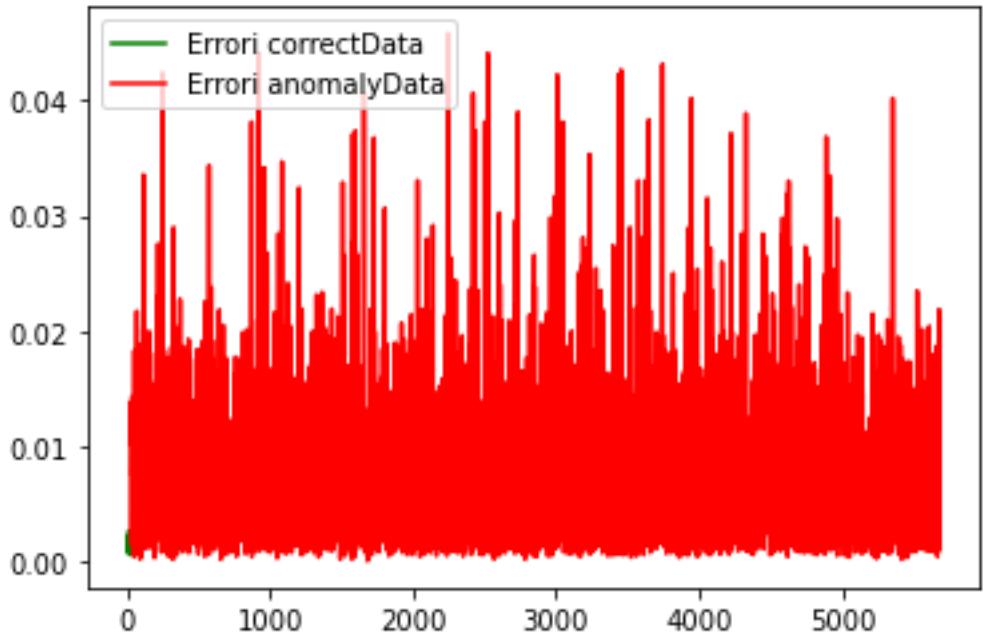
```
my_init = keras.initializers.glorot_uniform(seed=1)
autoenc = keras.models.Sequential()
autoenc.add(keras.layers.Dense(input_dim=i*j, units=100, activation='tanh', kernel_initializer=my_init))
autoenc.add(keras.layers.Dense(units=2400,activation='tanh', kernel_initializer=my_init))
autoenc.add(keras.layers.Dense(units=100,activation='tanh', kernel_initializer=my_init))
autoenc.add(keras.layers.Dense(units=50,activation='tanh', kernel_initializer=my_init))
autoenc.add(keras.layers.Dense(units=100,activation='tanh', kernel_initializer=my_init))
autoenc.add(keras.layers.Dense(units=2400,activation='tanh', kernel_initializer=my_init))
autoenc.add(keras.layers.Dense(units=i*j,activation='tanh', kernel_initializer=my_init))
autoenc.compile(optimizer='adam', loss='mse', metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2,patience=1, min_lr=0.000001)
```

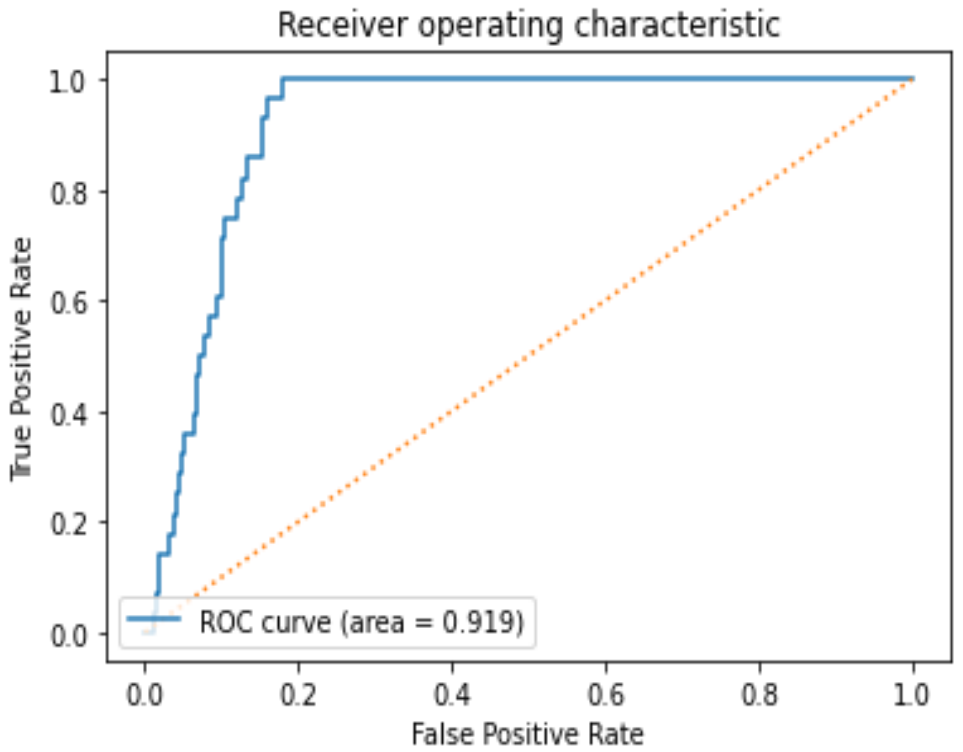






PCA


Errori correctData
Errori anomalyData
0.04
0.03
0.02
0.01
0.00
0
1000
2000
3000
4000
5000


Receiver operating characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.919)
0.0
0.2
0.4
0.6
0.8
1.0

PCA

```
y = dataFrame["class"]
X = dataFrame.drop("class",axis=1).values
anomaly_detection=loadM("BestAnomayModel")
pred=anomaly_detection.inverse_transform(anomaly_detection.transform(X))
err= mean_squared_err(X,pred)
y_pred=(np.array(thresholdErr(err,0.0001)))
accuracy_anomaly=accuracy_score(y,y_pred)*100
print(str(np.round(accuracy_anomaly,3))+" % ")
```

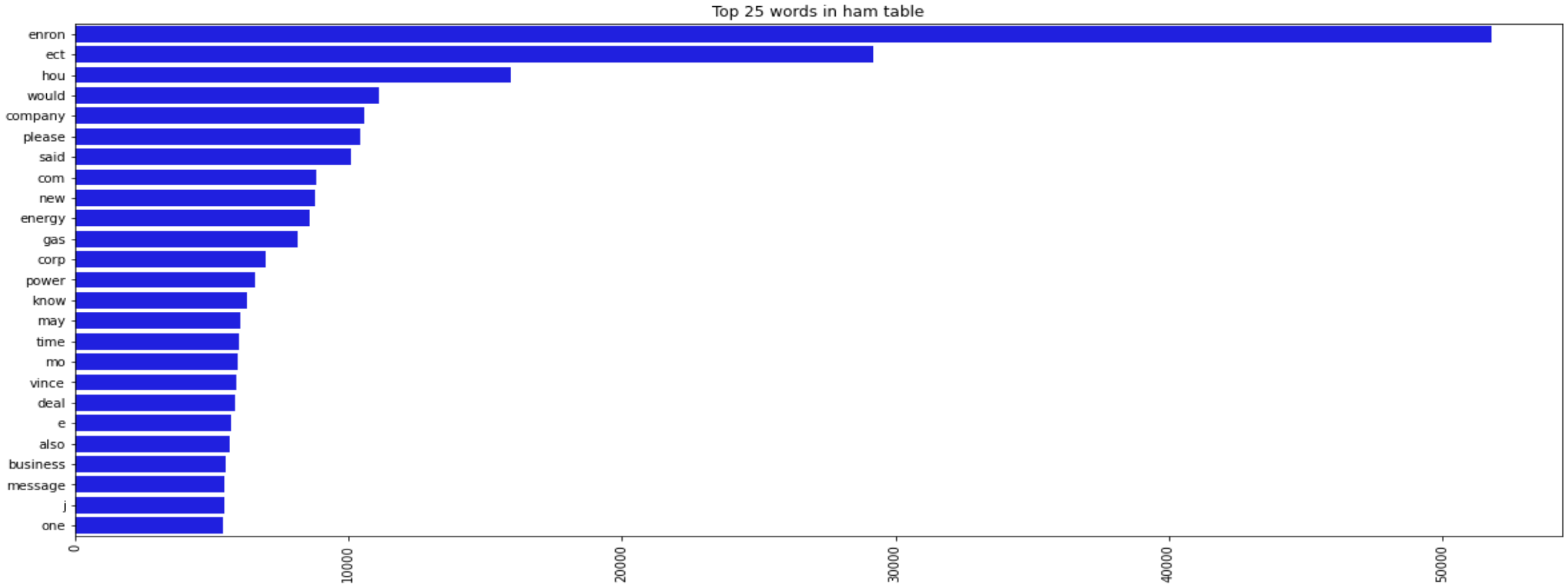
```
99.468 %
```

CARICAMENTO DEL DATASET TESTUALE

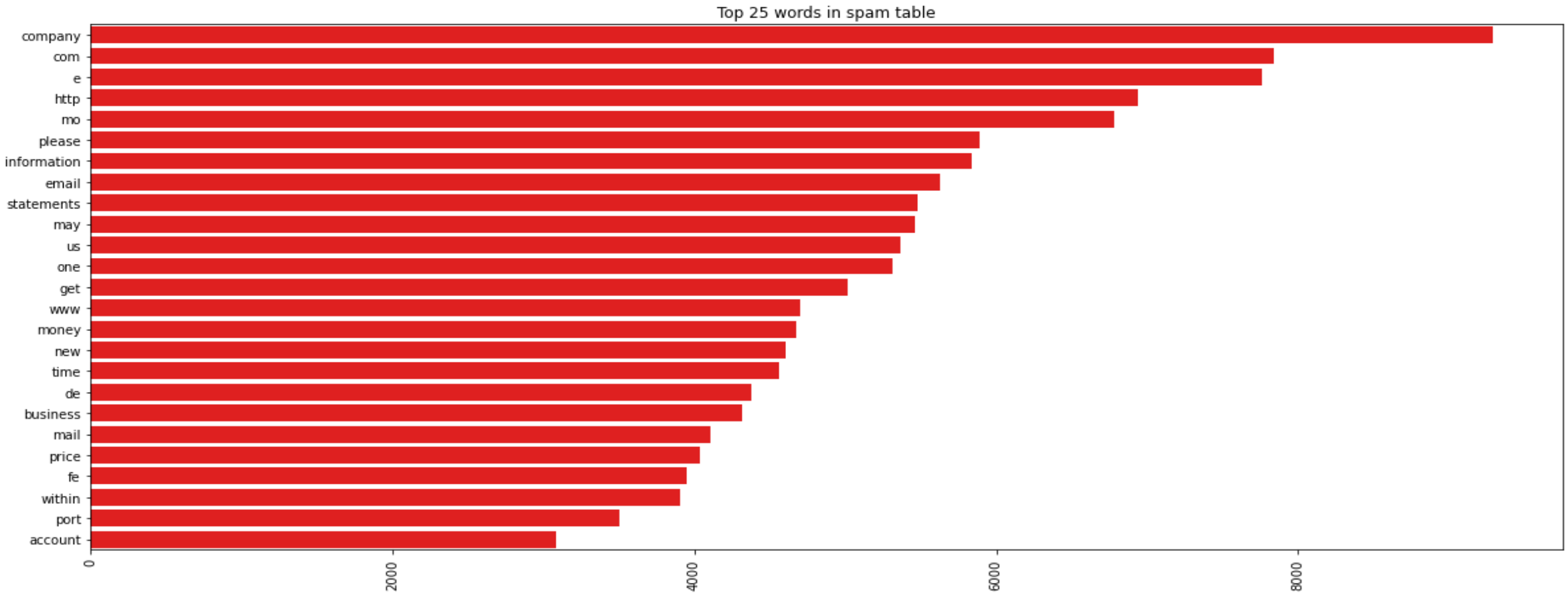
	text	target
0	Subject: fw : new generation report for june 2...	0
1	Subject: confirmation of meeting\nvince : than...	0
2	Subject: re : associate prc - input needed\ni ...	0
3	Subject: re : gwen koepke\nanne ,\nthanks for ...	0
4	Subject: become a stud in bed\nfact :\nin a po...	1

	text	target
0	[fw, new, generation, port, june, guys, intest...	0
1	[confirmation, meetingvince, thanks, introduci...	0
2	[associate, prc, input, neededi, houston, th, ...	0
3	[gwen, koepkeanne, thanks, contacting, matter,...	0
4	[become, stud, bedfact, poll, conducted, congl...	1

TOP 25


Top 25 words in ham table
enron
ect
hou
would
company
please
said
com
new
energy
gas
corp
power
know
may
time
mo
vince
deal
e
also
business
message
j
one
0
10000
20000
30000
40000
50000

TOP 25


Top 25 words in spam table
company
com
e
http
mo
please
information
email
statements
may
us
one
get
www
money
new
time
de
business
mail
price
fe
within
port
account
0
2000
4000
6000
8000

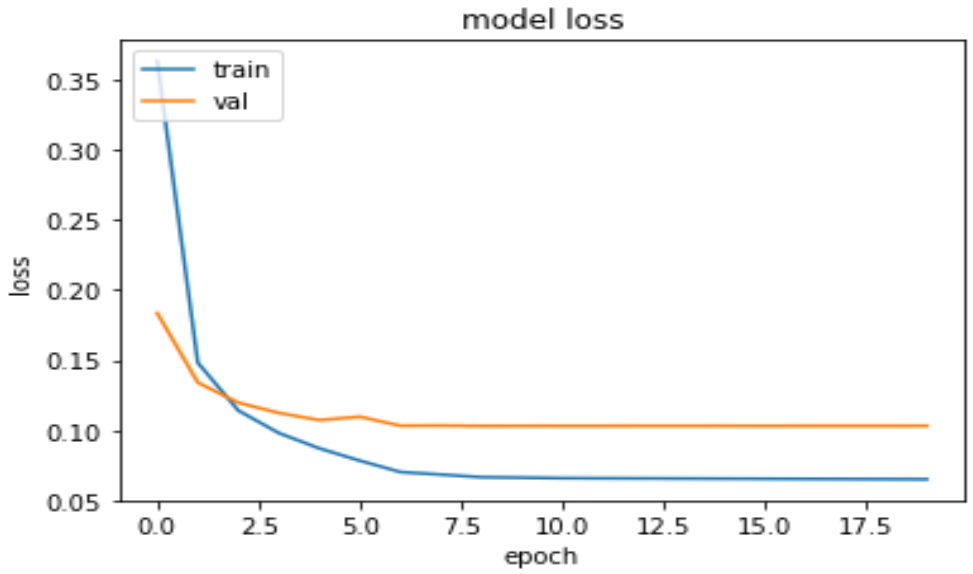
EMBEDDING 1

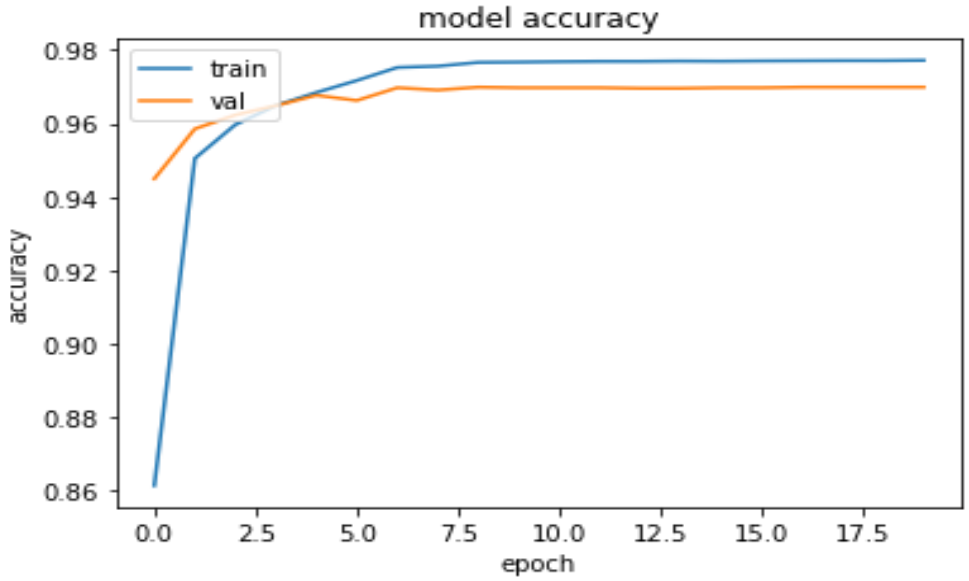
- Si crea un embedding per la tabella nel seguente modo:
  1) Creo un dizionario contenente al max le 200 parole più frequenti per entrambe le tipologie di mail;
  2) Creo un vettore che, per ogni file, analizza le parole e la converte con l'indice che la rappresenta nel dizionario, se questa è presente, altrimenti col valore zero;
  Otterrò, per il dataset in esame, una rappresentazione vettoriale di grandezza 305.

	text	target
0	[1, 0, 0, 1, 0, 1, 0, 1, 6, 0, 0, 0, 3, 0, 0, ...	0
1	[0, 0, 0, 0, 2, 2, 0, 0, 0, 2, 1, 0, 1, 0, 1, ...	0
2	[18, 10, 4, 2, 0, 3, 0, 0, 0, 0, 0, 2, 0, 0, 0...	0
3	[2, 1, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0
4	[0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, ...	1

RETE NEURALE

```
def create_network():
    opt = keras.optimizers.Adam(learning_rate=0.0001)
    model = Sequential()
    model.add(keras.layers.Dense(128, activation='relu',input_dim=305))
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dense(32, activation='relu'))
    model.add(layers.Dense(10, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))
    model.compile(optimizer=opt,loss='binary_crossentropy', metrics=['accuracy'])
    return model
```



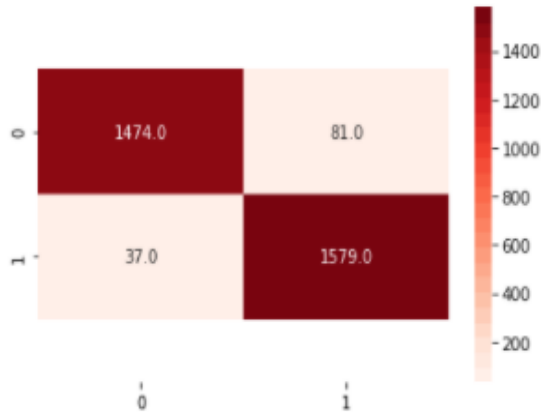


TUNING DEI MODELLI SVC,ADABOOST E KNN

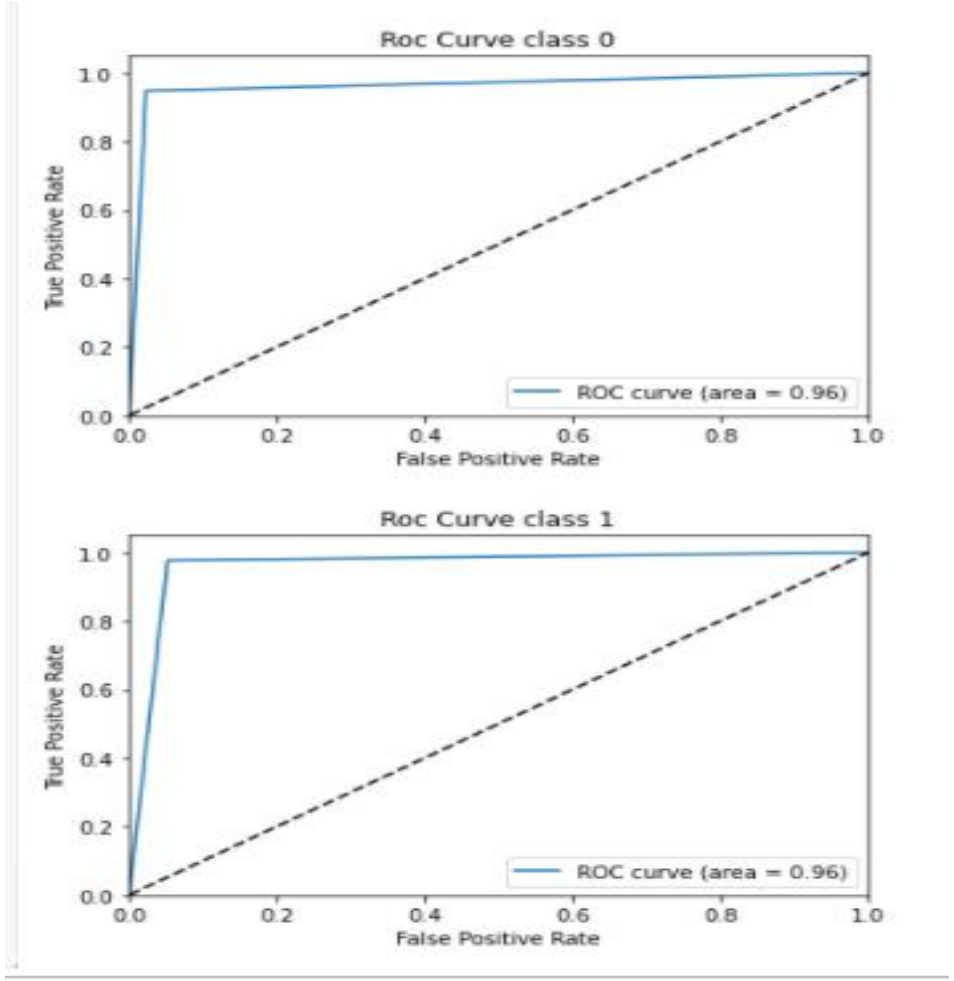
```
Training modelli...
AdaBoost...
Best Param: {'learning_rate': 1.0, 'n_estimators': 500}
Tuning SVC...
Best Param: {'C': 200, 'gamma': 'scale', 'kernel': 'rbf'}
Tuning KNN...
Best Param: {'metric': 'euclidean', 'n_neighbors': 9, 'weights': 'distance'}
Fatto!
```

ADABOOST

```
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.96278776 0.95868811 0.95425868 0.9615142  0.96088328 0.95425868
 0.96277603 0.96119874 0.95615142 0.95741325]
Ottenendo un'accuratezza media di 0.959 con deviazione standard di 0.003
La migliore accuratezza e' 0.963 ottenuta nello split numero 1
Salvo il modello ottenuto nello split numero 1
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 1
```

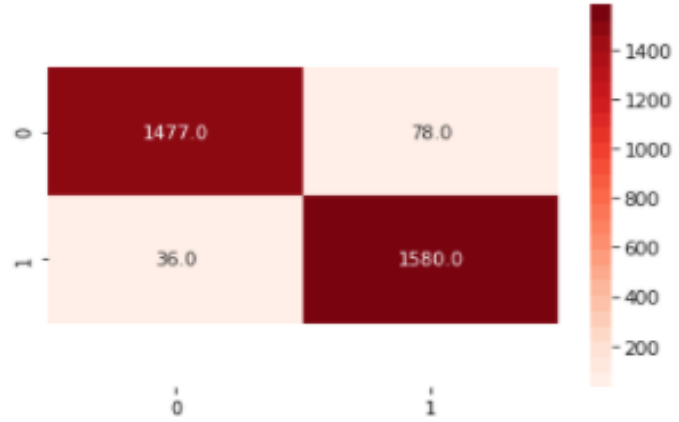


	precision	recall	f1-score	support
0	0.98	0.95	0.96	1555
1	0.95	0.98	0.96	1616
accuracy			0.96	3171
macro avg	0.96	0.96	0.96	3171
weighted avg	0.96	0.96	0.96	3171

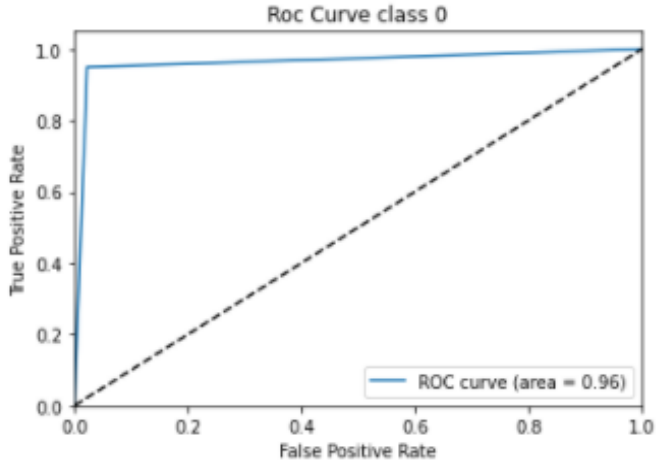


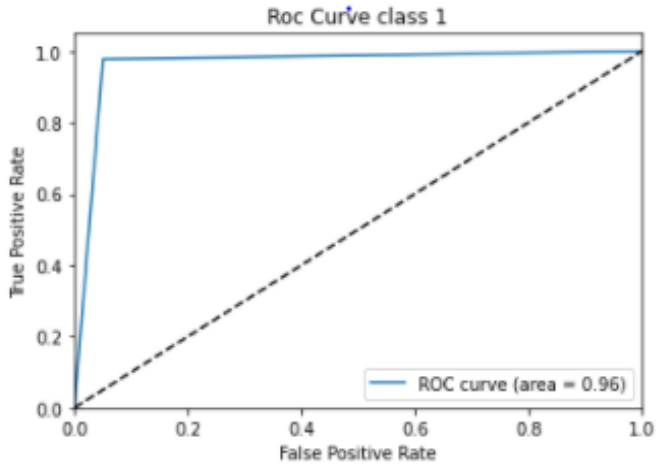
SVC

```
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.9640492  0.95931883 0.95457413 0.96214511 0.96214511 0.9533123
 0.96246057 0.96119874 0.95835962 0.95930599]
Ottenendo un'accuratezza media di 0.96 con deviazione standard di 0.003
La migliore accuratezza e' 0.964 ottenuta nello split numero 1
Salvo il modello ottenuto nello split numero 1
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 1
```



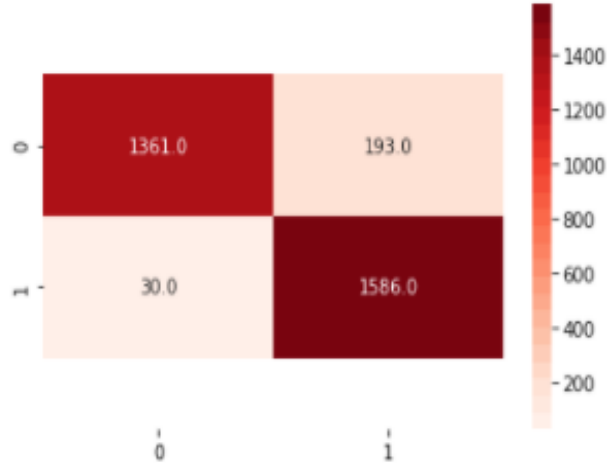
	precision	recall	f1-score	support
0	0.98	0.95	0.96	1555
1	0.95	0.98	0.97	1616
accuracy			0.96	3171
macro avg	0.96	0.96	0.96	3171
weighted avg	0.96	0.96	0.96	3171



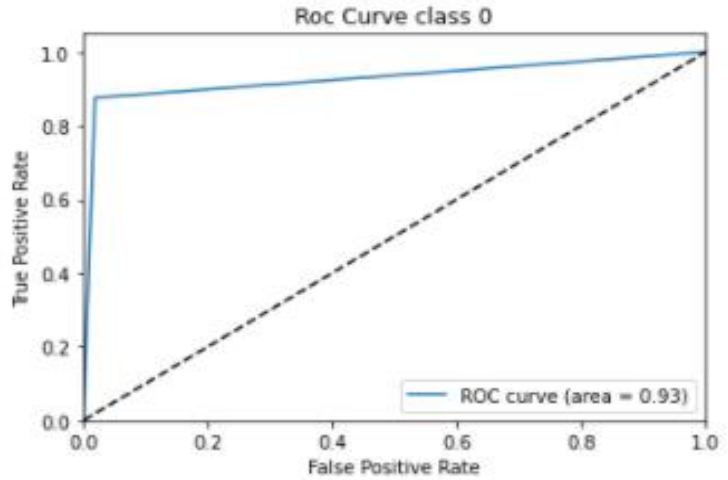


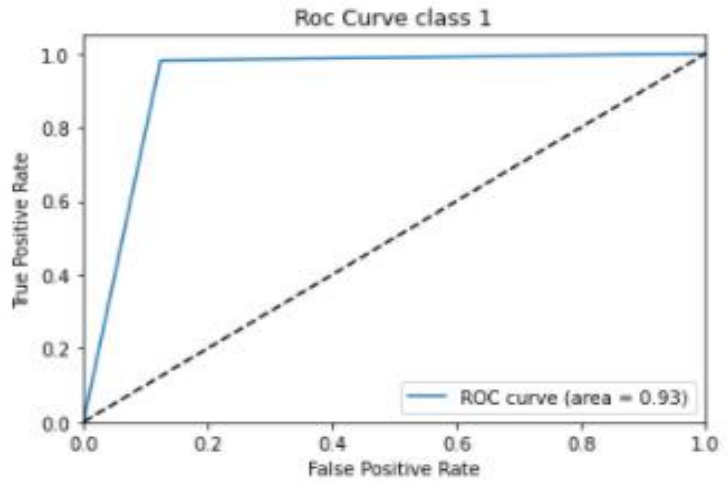
KNN

```
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.92210659 0.92336802 0.9148265  0.92681388 0.92807571 0.91514196
 0.92082019 0.92176656 0.929653   0.92586751]
Ottenendo un'accuratezza media di 0.923 con deviazione standard di 0.005
La migliore accuratezza e' 0.93 ottenuta nello split numero 9
Salvo il modello ottenuto nello split numero 9
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 9
```



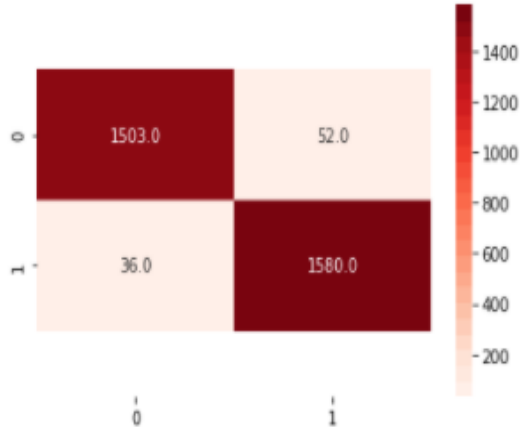
	precision	recall	f1-score	support
0	0.98	0.88	0.92	1554
1	0.89	0.98	0.93	1616
accuracy			0.93	3170
macro avg	0.93	0.93	0.93	3170
weighted avg	0.93	0.93	0.93	3170



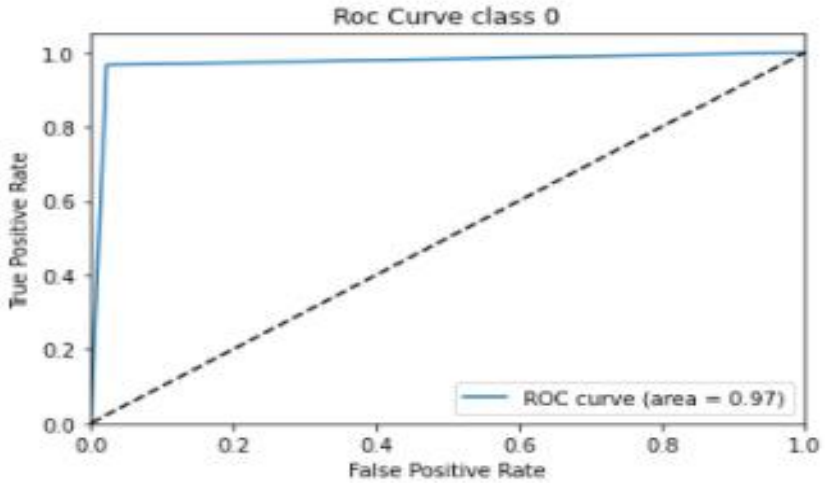


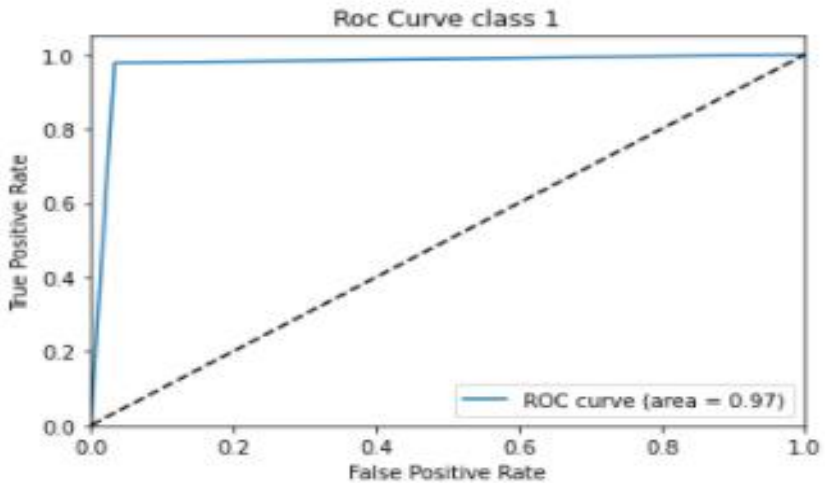
NN

Score per fold
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.9722484946250916, 0.968464195728302, 0.960567831993103, 0.970346987247467, 0.9646687507629395, 0.9596214294433594, 0.9706624746322632, 0.9662460684776306, 0.9630914926528931, 0.9649842381477356]
Ottenendo un'accuratezza media di 0.966 con deviazione standard di 0.004
La migliore accuratezza e' 0.972 ottenuta nello split numero 1
Salvo il modello ottenuto nello split numero 1
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 1



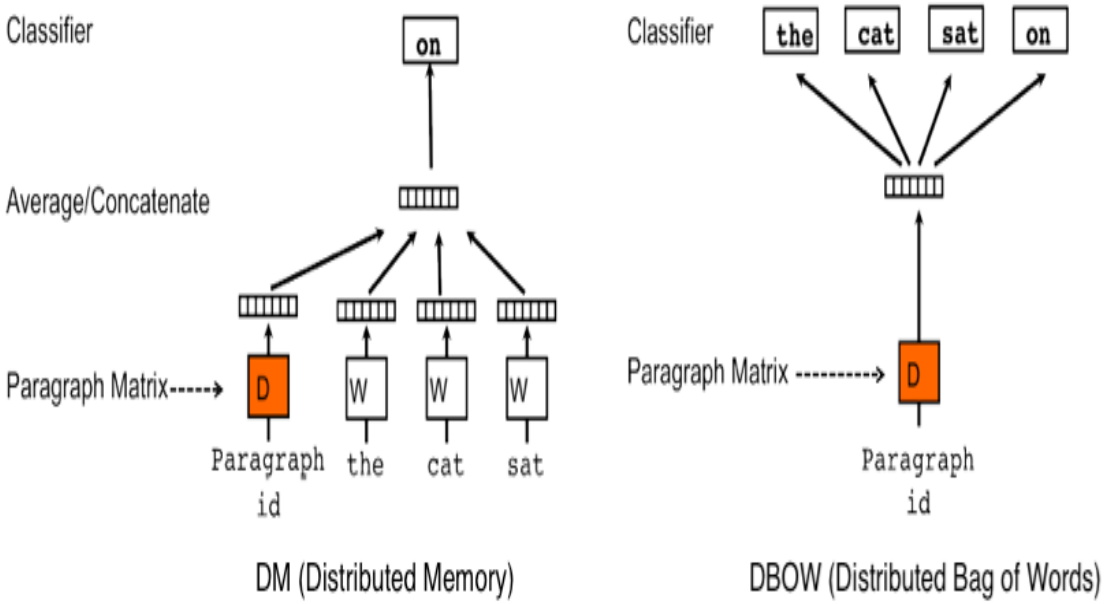
	precision	recall	f1-score	support
0	0.98	0.97	0.97	1555
1	0.97	0.98	0.97	1616
accuracy			0.97	3171
macro avg	0.97	0.97	0.97	3171
weighted avg	0.97	0.97	0.97	3171





EMBEDDINGS TRAMITE DOC2VEC

- Crea una rappresentazione numerica di un intero documento (o paragrafo, nel nostro caso di un tweet) a prescindere dalla sua lunghezza. Il principio usato è semplice ed intelligente: si fa uso del modello word2vec e si aggiunge un altro vettore, detto Paragraph ID



Si utilizzerà l'algoritmo PV-DBOW, con un vettore di taglia 500 per rappresentare il documento. La predizione avrà una finestra di 2 parole e si utilizzeranno 10 epoche di train.

EMBEDDING TRAMITE DOC2VEC

```
]: df_doc2vec=pd.read_pickle("/home/emiliocasella/Scrivania/proj204898/txt/cleanwords")
   words=df_doc2vec.text.to_list()
   target=df_doc2vec.text.to_numpy()
   documents = [TaggedDocument(words[i], target[i]) for i in range(len(target))]
   model = gensim.models.Doc2Vec(documents, vector_size=500, window=2,epochs=10, min_count=1, workers=core
   model.save('d2vec'+'.model')
   print("Fatto!")
```

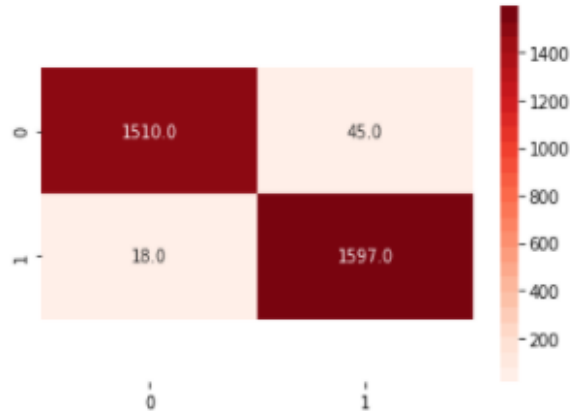
```
Fatto!
```

```
MyDoc2vec().transform(df_doc2vec,['text'],"d2vec.model")
df_doc2vec.head()
```

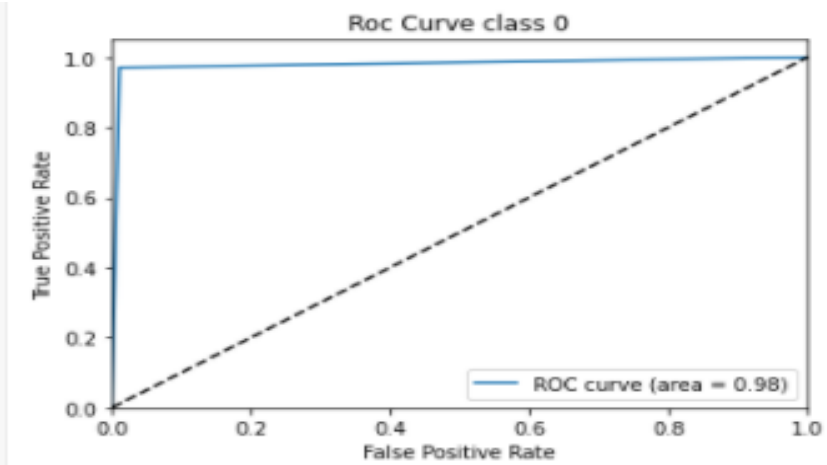
	text	target
0	[0.17600273, -0.33752394, 0.10832454, -0.03667...	0
1	[-0.2561136, 0.08447421, 0.090536326, -0.05035...	0
2	[-0.8033857, 0.3208517, 0.7965166, -0.14490508...	0
3	[-0.3176085, 0.45610327, 0.20239797, 0.3979420...	0
4	[-0.74756914, 0.009872089, -0.020475907, 1.624...	1

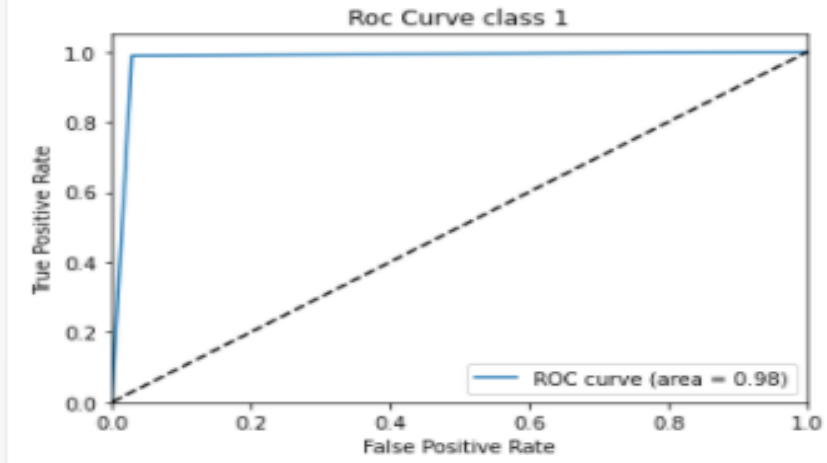
NN + DOC2VEC

```
Score per fold
L'accuratezza per ogni sacca della cross validation è rappresentata dal seguente vettore
[0.9757174253463745, 0.9728792309761047, 0.960567831993103, 0.9801262021064758, 0.9675078988075256,
0.969085156917572, 0.9637224078178406, 0.967823326587677, 0.9706624746322632, 0.9709779024124146]
Ottenendo un'accuratezza media di 0.97 con deviazione standard di 0.005
La migliore accuratezza e' 0.98 ottenuta nello split numero 4
Salvo il modello ottenuto nello split numero 4
Si mostra ora la matrice di adiacenza ottenuta sul test set dello split numero 4
```



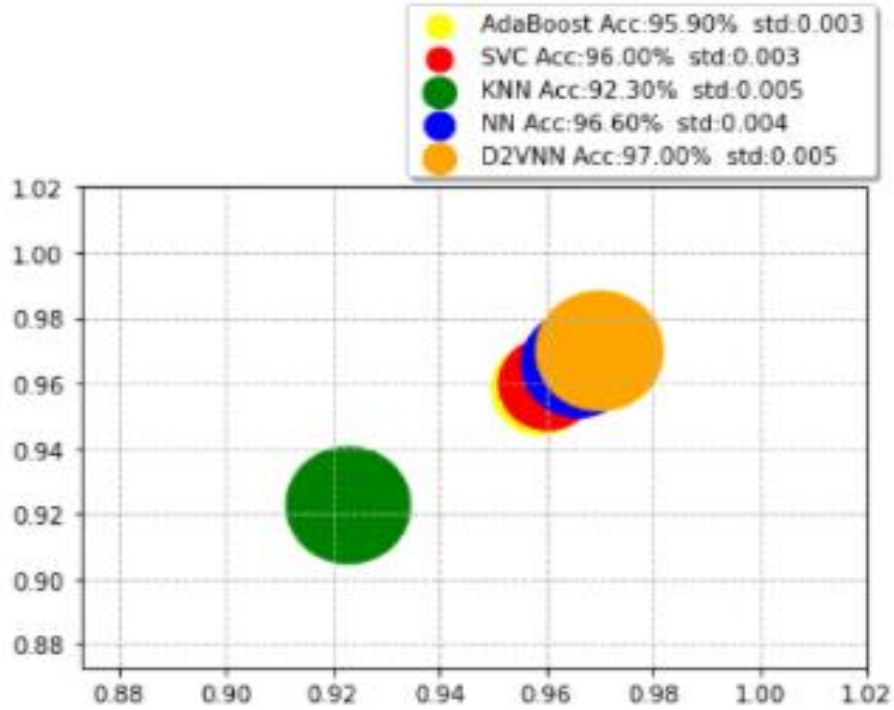
	precision	recall	f1-score	support
0	0.99	0.97	0.98	1555
1	0.97	0.99	0.98	1615
accuracy			0.98	3170
macro avg	0.98	0.98	0.98	3170
weighted avg	0.98	0.98	0.98	3170





ACCURATEZZA SULLA 10-FOLD CROSS-VALIDATION

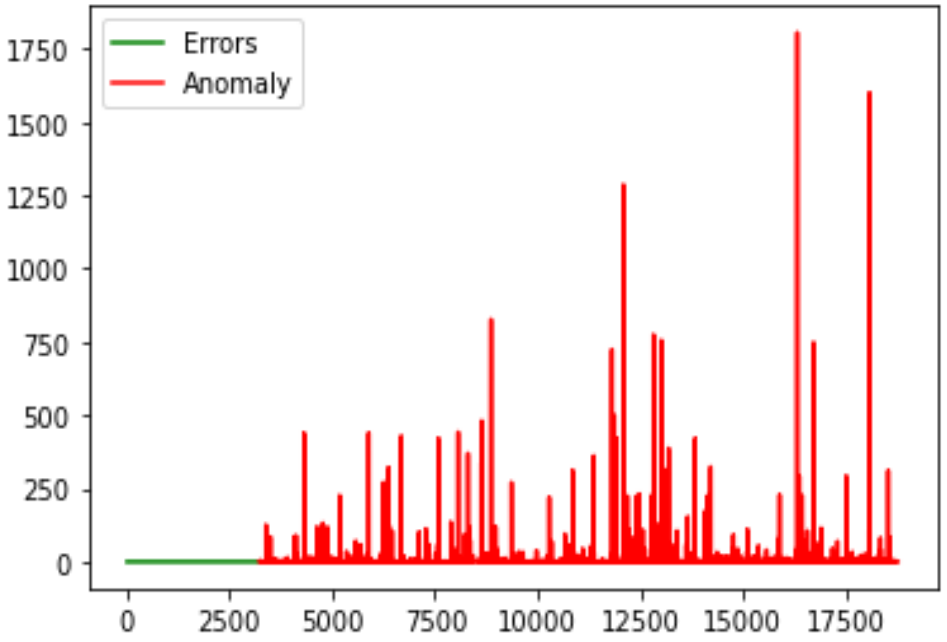
	accuracy_mean	std
AdaBoost	0.959	0.003
SVC	0.960	0.003
KNN	0.923	0.005
NN	0.966	0.004
D2VNN	0.970	0.005

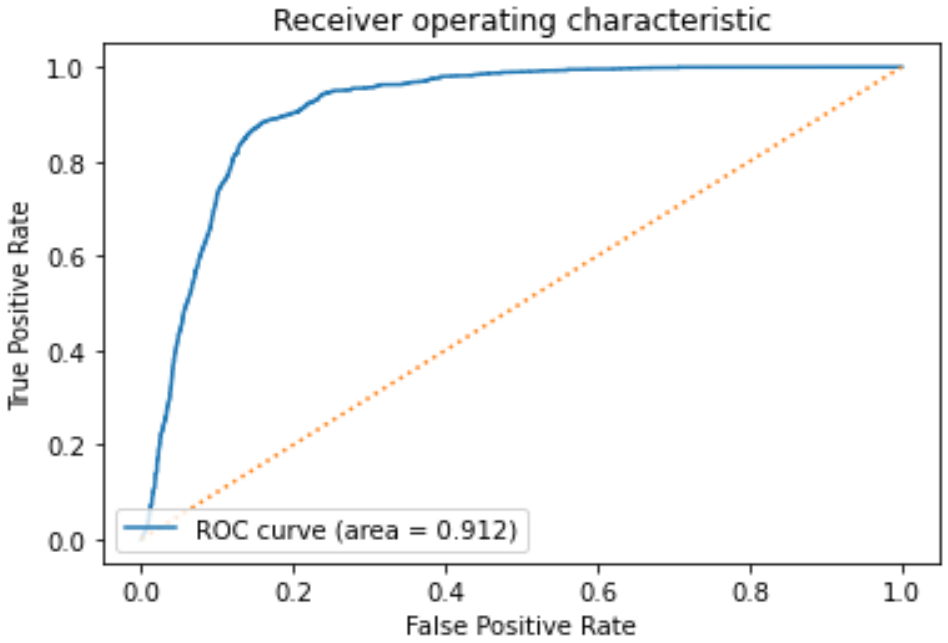


ANOMALY DETECTION

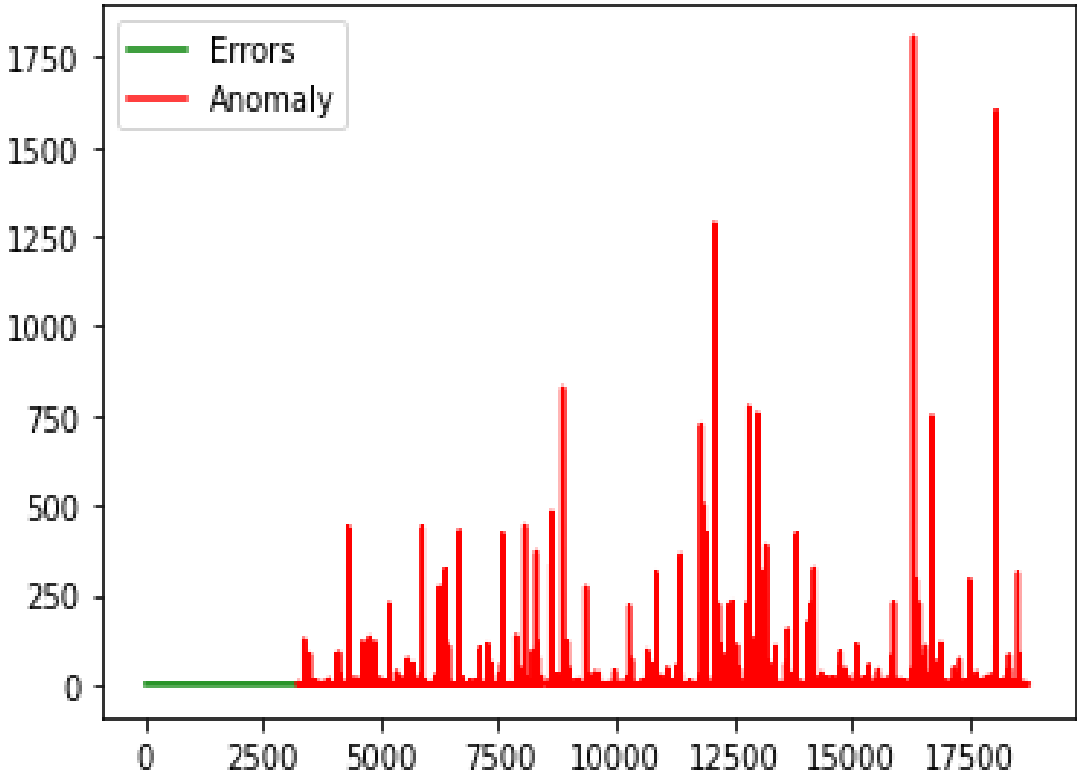
PCA EMBEDDING 1

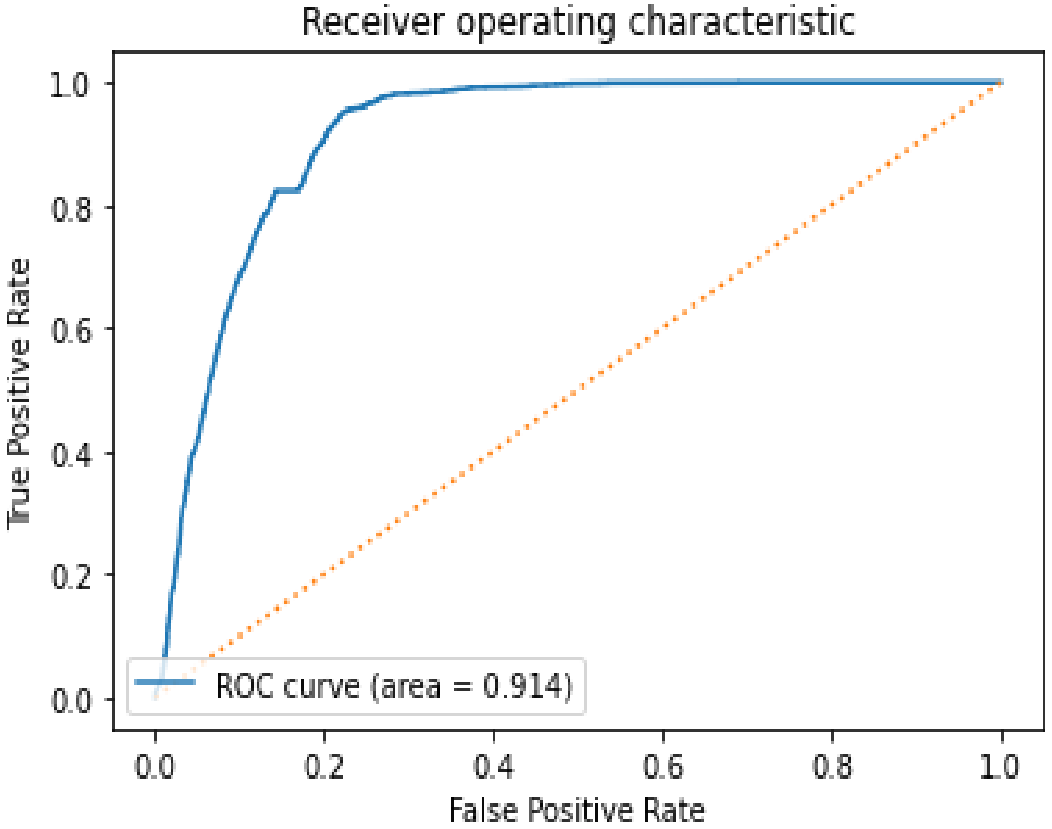
- La migliore configurazione è con 256 features. Il modello non commette errore, quasi nullo sulla classe corretta, presentando un buon risultato. La curva di ROC conferma tutto questo, mostrando un risultato altamente accurato.





PCA EMBEDDING DOC2VEC





FINE

Grazie per l'attenzione.