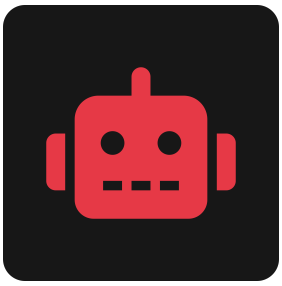
Classificazione di Topics in base ad una domanda: trained on Yahoo Answers

Giacomo Paolocci, Raffaele Lorusso

Febbraio 2024



1 Introduzione

Il progetto presentato ha come obiettivo la presentazione di una dimostrazione del modello di Machine Learning *classification*. Tale presentazione vede l'assegnazione di topics di Yahoo Answers all'input proposto all'AI. Infine un CSP problem sarà usato per eventualmente filtrare le assegnazioni effettuate.

2 Materiali e Metodi

- come materiale usato per allenare il modello Machine Learning è stato usato e modificato un dataset che raggruppa 1.4M di domande e risposte di Yahoo! Answers, si trova al seguente link
- Per lo sviluppo e allenamento del modello Machine Learning sono stati usati gli strumenti della libreria python *sklearn* per l'allenamento e il file *NLTKVectorizer.py* per vettorializzare il contenuto del dataset e l'input per la previsione, tale file è fornito dal tutorial a questo link, ed è stato trattato come materiale di riferimento per e durante lo svolgimento del progetto
- Per l'interfaccia grafica è stato usato HTML, CSS e Typescript affiancato dal framework *Tauri*

- Per connettere interfaccia grafica con il modello ML è stato usato il web framework in python *Bottle*, il quale crea un server locale in cui far eseguire le previsioni del modello

3 Esperimenti e risultati

3.1 Legenda per i topic

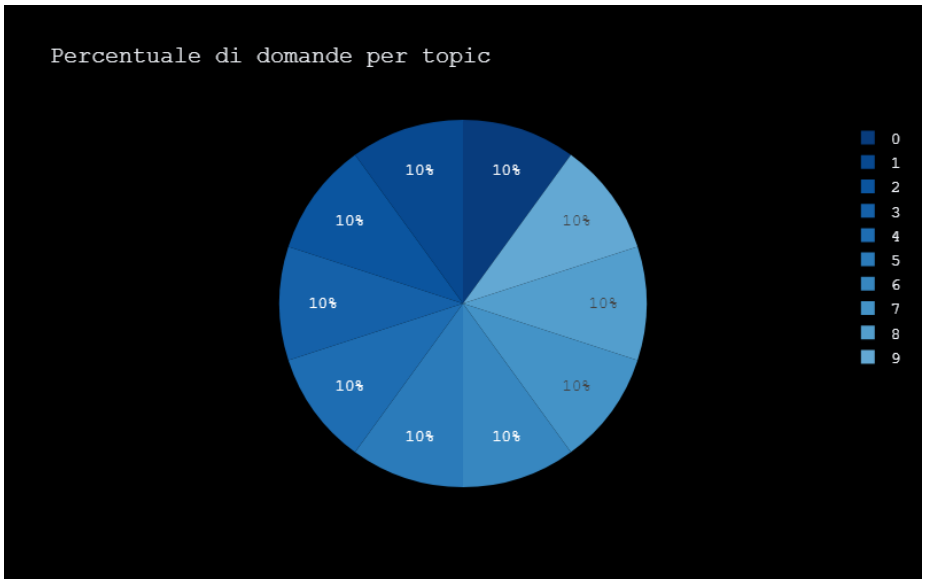
- 0: **Society & Culture**
- 1: **Science & Mathematics**
- 2: **Health**
- 3: **Education & Reference**
- 4: **Computers & Internet**
- 5: **Sports**
- 6: **Business & Finance**
- 7: **Entertainment & Music**
- 8: **Family & Relationships**
- 9: **Politics & Government**

3.2 Modifica del dataset

Prima dell'allenamento del modello abbiamo ritenuto opportuno modificare il dataset in modo da unire il campo *question_title* con il campo *question_conten*t così da uniformare il materiale per l'allenamento.

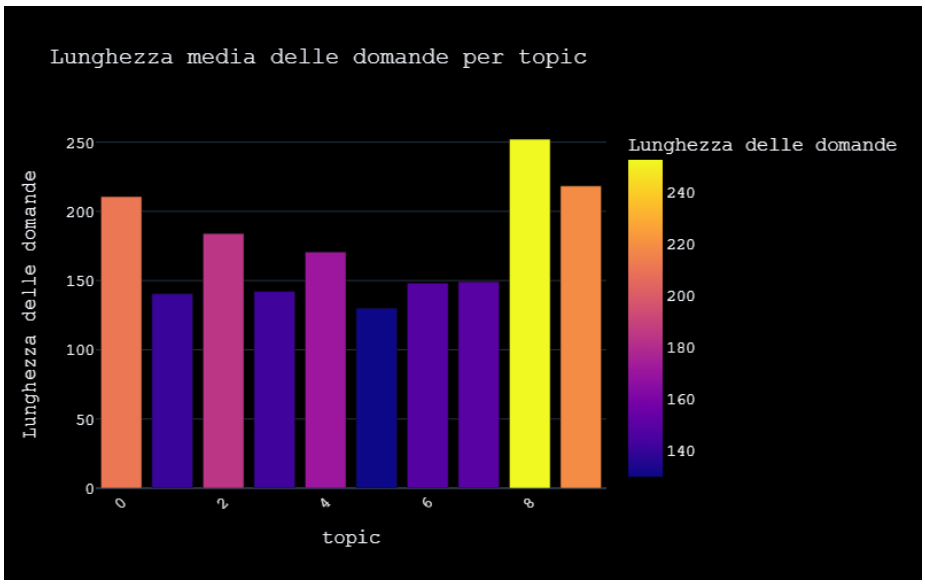
3.3 Percentuale di domande per topic

Abbiamo iniziato a valutare il nostro dataset vedendo se il numero di domande per topic è ben distribuito.



Notiamo quindi che il dataset ha una equa distribuzione di domande per topic, cosa più che ottima

3.4 Lunghezza di domande per topic



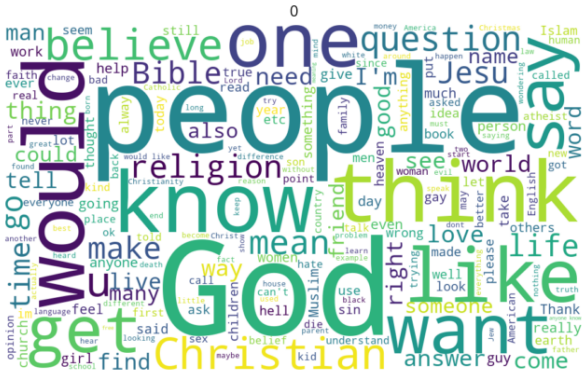
Notiamo che la lunghezza delle domande è ben distribuita per tutti i topic, tranne per il topic 0 (Society & Culture), 8 (Family & Relationships) e 9 (Politics

& Government), che hanno una lunghezza media delle domande leggermente più lunga rispetto agli altri topic.
Abbiamo comunque deciso di non agire sul bilanciamento, essendo che rispetto alla media c'è poca differenza, inoltre, senza un criterio di eliminazione ottimo, si sarebbe solo andato ad intaccare la qualità dei dati.

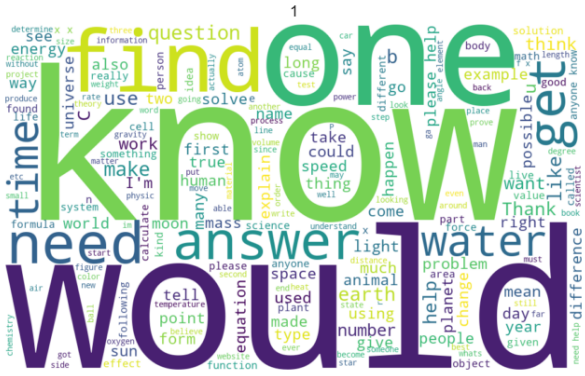
3.5 WordCloud

Il dataset, per ogni topic, conterrà una varietà di parole inutili per l'apprendimento, in quanto possono essere comuni a più topics in modo uniforme e/o essere semplici particelle grammaticali, usando WordCloud abbiamo identificato quelle più prevalenti e aggiunte ad una lista di *stopwords*, che verrà passata al primo stage della pipeline per eliminare dall'apprendimento le parole elencate.

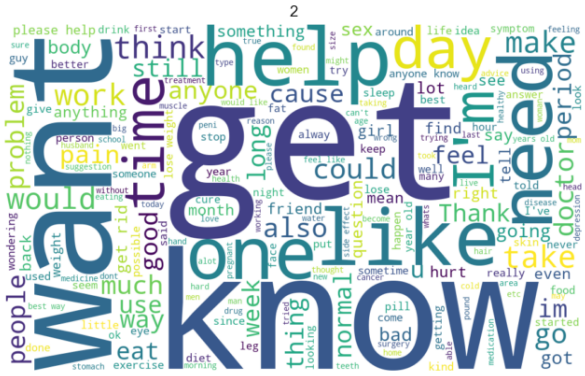
Seguono i grafici Wordcloud per ogni topic:



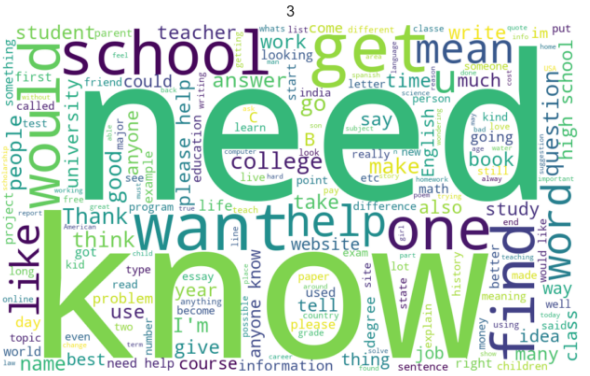
(a) Society & Culture



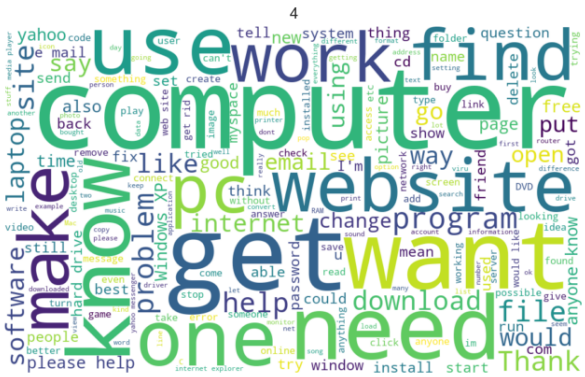
(b) Science & Mathematics



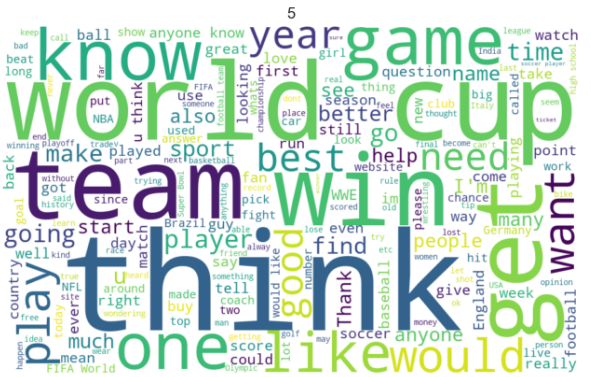
(c) Health



(d) Education & Reference

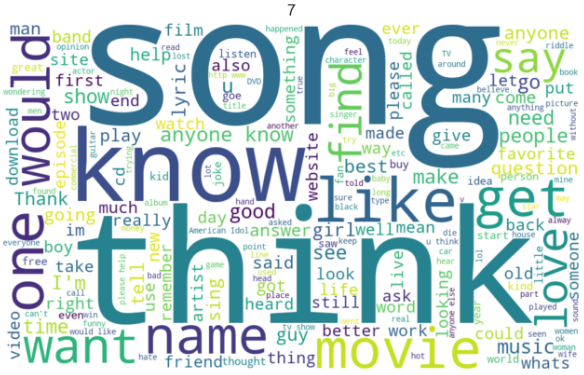


(e) Computers & Internet



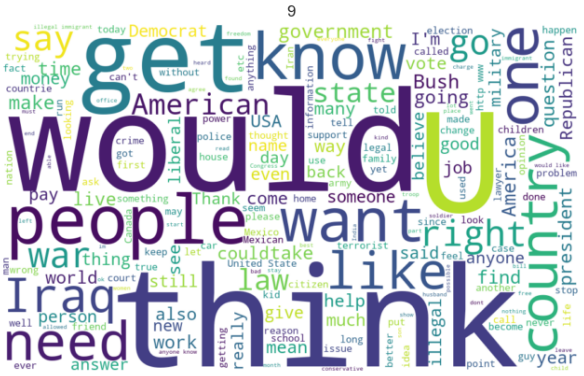
(f) Sports

(a) Business & Finance



(b) Entertainment & Music

(c) Family & Relationships



(d) Politics & Government

Notiamo che le parole più comunemente usate, che però non rispecchiano il topic, sono:

- think
- would
- get
- want

Quindi sono state aggiunte alla lista di stopwords.

3.6 Dataset splitting

In quanto a splitting del dataset, esso è stato già reso disponibile all'origine diviso in un dataset di testing, e di training.

3.7 Training e prediction

Per l'allenamento e previsione abbiamo creato una pipeline la quale effettuerà 2 operazioni:

- 1. vettorializzazione dei dati, trasformando parole in de facto vettori numerici, in modo da essere commestibili per il modello, inoltre tali parole verranno eliminate se presenti nell'elenco *stopwords* passata all'operazione
- 2. allenamento: Anche se il LinearSVC è il modello ottimale, come verrà illustrato più avanti, è stato usato un modello basato su Logistic Regression, esso durante il training ha ricevuto il dataset training.csv vettorializzato dallo step precedente.

Abbiamo allenato e studiato entrambi i modelli, abbiamo notato che in accuratezza abbiamo in entrambe il 67%

3.8 GridSearch

Attraverso il metodo chiamato *GridSearchCV* si può effettare una ricerca brutale dei parametri ottimali che permettono al modello di avere meno errori. Per motivi di non disponibilità di dispositivi abbastanza veloci ne abbiamo condotta una molto ristretta a scopo esemplificativo, il risultato è stato di voler preferire il modello *RandomForest* per l'allenamento, ma la limitatezza di tale ricerca sarà dimostrata nei paragrafi successivi.

3.9 Training e Prediction con LinearSVC

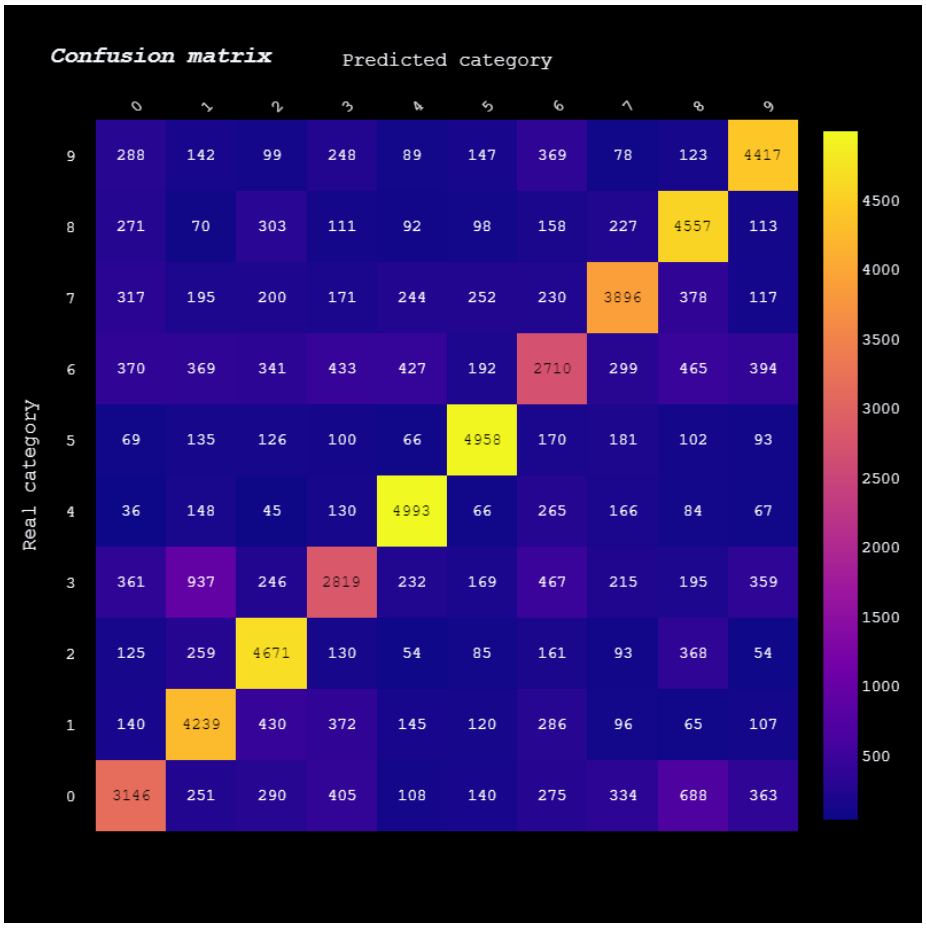
3.9.1 Fit e Predict

```
    %%time
    pipeline.fit(X_train, y_train)
    #%%
    %%time
    y_pred = pipeline.predict(X_test)
    #y_pred = model.predict(X_test)
    print(mean_squared_error(y_test, y_pred))

```

Come errore abbiamo: 6.25
Considerabile come errore abbastanza alto

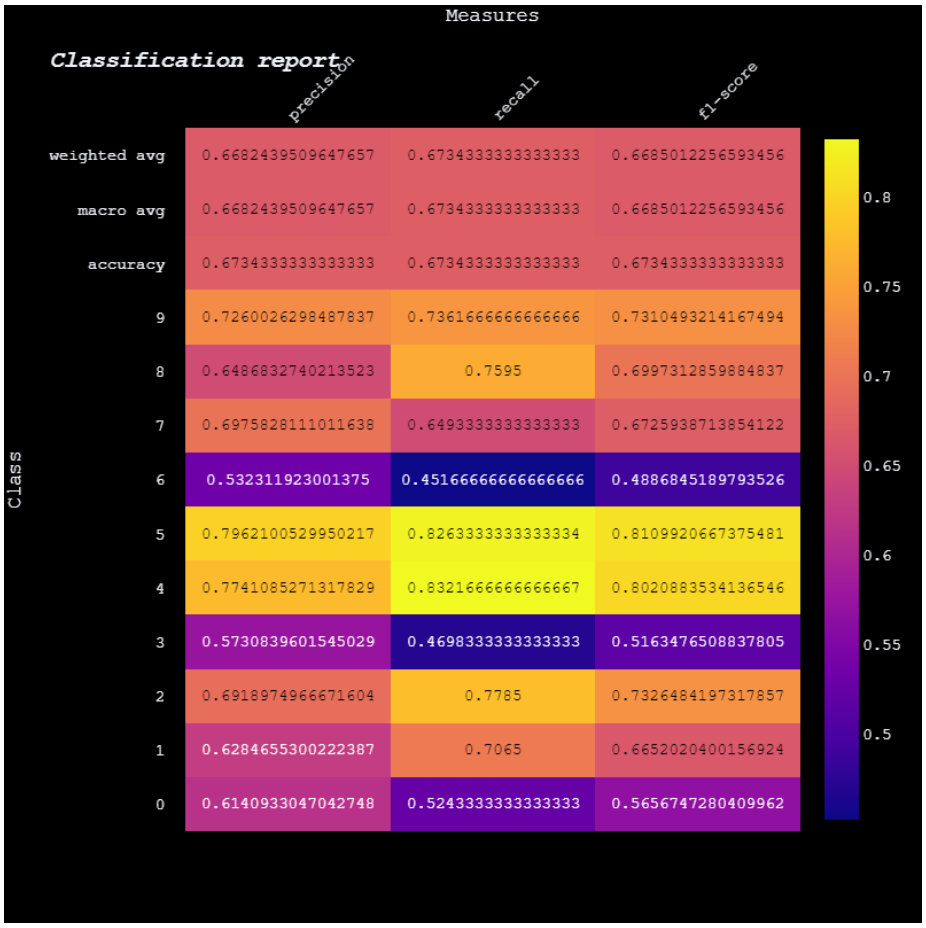
3.9.2 Risultati

Sono mostrate di seguito le confusion e error matrixes dopo aver adottato *LinearSVC*:



- Totale predizioni: 60000
- Predizioni giuste: 40406

Nel grafico successivo andiamo a vedere la percentuale di accuratezza e l'f1-score per ogni topic e in media



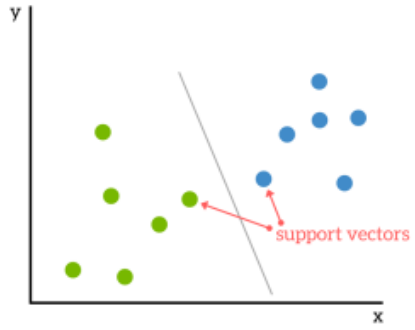
Mettendo a paragone con i grafici del training del Logistic Regression nella sezione 3.11, si può notare un leggero miglioramento del modello

	text	real category	predicted category	text length
47774	Is this ok for a personal statement? I'm in year 11 and applying for college soon.? Personal Statement – Kadie Bark My name Is Kadie Bark, I'm 15 years old and I am currently attending The Sh...	3	3	4001
30377	What do you think of this joke? read on! You want regret it if you read it, trust me. what do you think?? Pocket Taser Stun Gun, a great gift for the wife. This was submitted by a guy who purcha...	7	2	3972
16752	from the washington times maybe we should adopt their ilegal immigration policies go protest your own land? The Mexican solution FRANK J. GAFFNEY JR. By Frank J. Gaffney, Jr. April 4, 2006 The C...	9	9	3971
58108	Love one another as I love you. Only then, will people know that you are my disciples. Why argue? If Jesus is G– or of G–, does it change your Faith in Jesus? If the Holy Spirit is G– or of G-...	0	0	3962
52192	CALLING ALL HANDICAPPERS...The Belmont Stakes-June 10-2006? Who do u like to win the Belmont Stakes to be run in New York on June 10-2006 ? Kentucky Derby (G1) runner-up Bluegrass Cat and Peter Pa...	5	5	3924

La tabella sopra raffigurata rappresenta quattro esempi random di testing e possiamo notare che solo 1 su 5 predizioni è risultata sbagliata

3.9.3 Come funziona il LinearSVC

I campioni, e nel nostro caso i token riconosciuti all'interno della frase, vengono allocati in uno spazio bidimensionale dove ogni asse rappresenta un topic. In questo spazio abbiamo una retta che separa i nostri token. I token più vicini alla nostra retta si chiamano *support vector*. La distanza di questi *support vectors* descrive il **margine**, ossia quanto la retta separi bene le due classi.



L'obiettivo dell'SVC è quello di **massimizzare** la larghezza del margine Nel nostro caso possiamo controllare 2 parametri:

- alpha

- C : Più grande è C, più avremmo campioni etichettati in modo errato. Allo stesso tempo il modello aumenterà robustezza diminuendo la precisione

```
            svm_clf = LinearSVC(C=1.0, verbose=2)

```

3.10 Training e Prediction con Logistic Regression

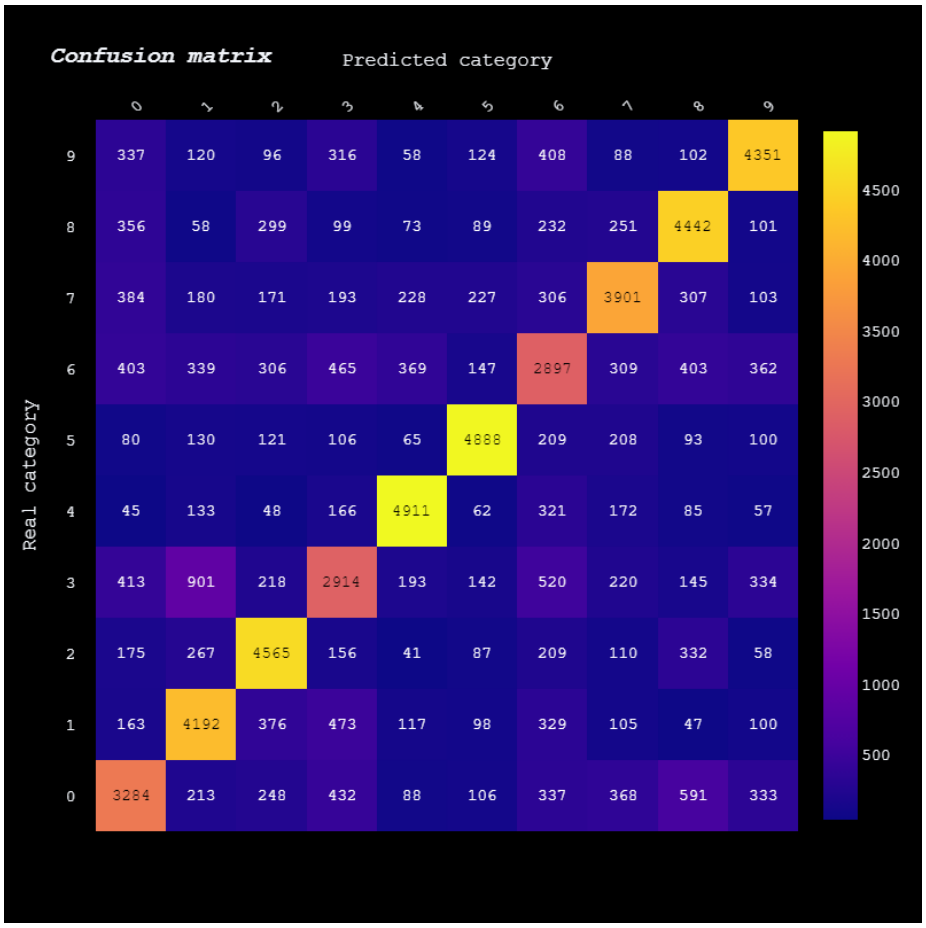
3.10.1 Fit e Predict

```
    %%time
    pipeline.fit(X_train, y_train)
    #%%
    %%time
    y_pred = pipeline.predict(X_test)
    #y_pred = model.predict(X_test)

    print(mean_squared_error(y_test, y_pred))

```

Come errore abbiamo: 6.30
Considerabile come errore abbastanza alto

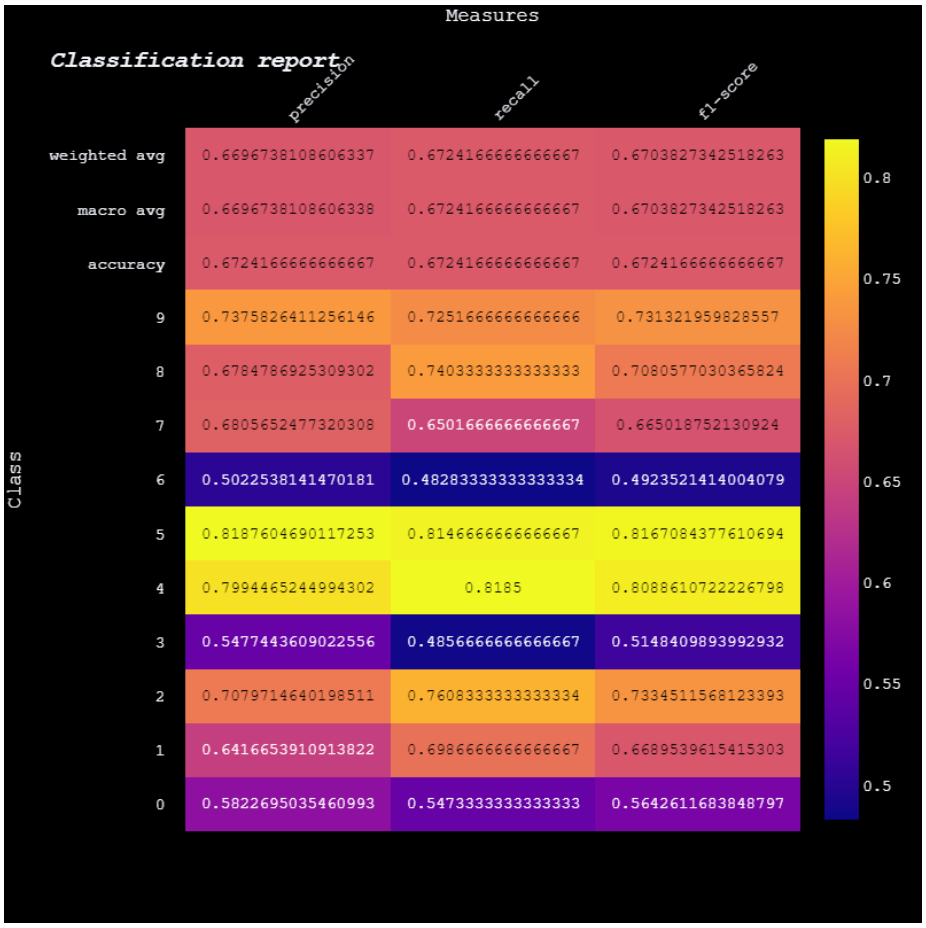
3.10.2 Risultati

Sono mostrate di seguito le confusion e error matrixes dopo aver adottato *Logistic regression*:



Nel grafico successivo andiamo a vedere la percentuale di accuratezza e l'f1-score per ogni topic e in media

- Totale predizioni: 60000
- Predizioni giuste: 40345



Notiamo che c'è quasi nessuna differenza con la LinearSVC

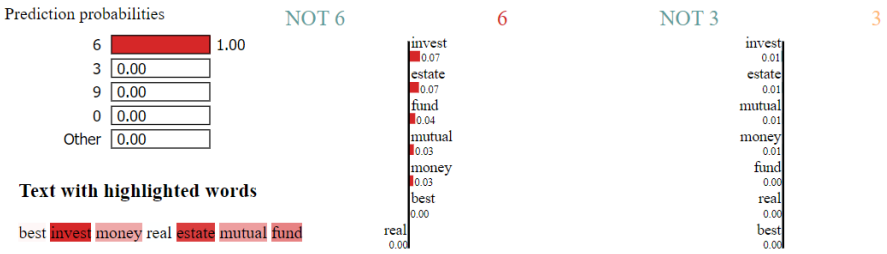
	text	real category	predicted category	text length
47774	Is this ok for a personal statement? I'm in year 11 and applying for college soon.? Personal Statement – Kadie Bark My name Is Kadie Bark, I'm 15 years old and I am currently attending The Sh...	3	3	4001
30377	What do you think of this joke? read on! You want regret it if you read it, trust me. what do you think?? Pocket Taser Stun Gun, a great gift for the wife. This was submitted by a guy who purcha...	7	2	3972
16752	from the washington times maybe we should adopt their ilegal immigration policies go protest your own land? The Mexican solution FRANK J. GAFFNEY JR. By Frank J. Gaffney, Jr. April 4, 2006 The C...	9	9	3971
58108	Love one another as I love you. Only then, will people know that you are my disciples. Why argue? If Jesus is G-- or of G--, does it change your Faith in Jesus? If the Holy Spirit is G-- or of G-...	0	0	3962
52192	CALLING ALL HANDICAPPERS...The Belmont Stakes-June 10-2006? Who do u like to win the Belmont Stakes to be run in New York on June 10-2006 ? Kentucky Derby (G1) runner-up Bluegrass Cat and Peter Pa...	5	5	3924

La tabella sopra raffigurata rappresenta quattro esempi di testing e possiamo notare che solo 1 su 5 predizioni è risultata sbagliata

3.10.3 Come funziona la Logistic Regression

Tramite la *Logistic Regression* prendiamo uno ad uno i topic e valutiamo se quel topic si addice al testo in input.
Essa è progettata per prevedere la probabilità che un'osservazione appartenga a una determinata classe (topic nel nostro caso).
Quindi la *Logistic Regression* è un modello statistico utilizzato per prevedere le probabilità di appartenenza a una classe, e viene comunemente utilizzato in problemi di classificazione binaria

```
        # Logistic Regression classifier
        lr_clf = LogisticRegression(C=1.0, solver="newton-cg",
    multi_class="multinomial", n_jobs=8, verbose=2)

```



Il topic 6 è **Business & Finance**

4 Scelta del modello

Confrontiamo l'accuratezza fra i due metodi:

- Logistic Regression è: 0.672
- LinearSVC è: 0.673

Invece, confrontando le precision, vediamo che su alcuni topic aumenta, mentre in altre diminuisce, prendendo in condiderazione la media standard, cioè la **macro avg** abbiamo:

- Logistic regression: 0.6697
- LinearSVC : 0.6682

Anche confrontando le confusion matrixes notiamo che il numero di outliers non sono molto differenti tra loro.

Vedendo che i due modelli sono molto simili abbiamo scelto la *logistic regression* essendo che offriva più metodi rispetto L'SVC.
Uno di questi è **pipeline.predict_proba()**, che utilizziamo per applicare il CSP e mostrare all'utente i risultati del suo input

5 Interfaccia grafica e usabilità

5.1 Python Server

Di seguito è il codice python del server:

```
from bottle import run, route, get, post, request, FormsDict,
    response
from joblib import load
from sklearn.pipeline import Pipeline as Pipeline
from json import dumps as jsondump
from CSP.CSP_main import run_csp

# decidere se avere cambiamenti del server in real-time mentre si
    modifica il .py (True)
```

```
DEBUG = False

# decidere se usare direttamente il modello presettato (True) o obbligare l'utente a caricare il modello
CUSTOM = False

model_pipeline: Pipeline = None if CUSTOM else load("../dump/model.joblib")


###################### metodi e classi di supporto
def list_to_dict(processing: list) -> dict:
    i = 0
    dictionary = dict()
    while i < len(processing):
        dictionary[i] = processing[i]
        i += 1
    return dictionary


###################### metodi per le richieste al server

@route("/load_model")
def load_server():
    global model_pipeline
    model_Pipeline = load('../dump/model.joblib')  # official model
    if model_pipeline is not None:
        return "model Loaded"


@post("/predict")  # predict ufficiale
def predict_post():
    global model_pipeline
    submitted = request.query["text"]  # ricevo il testo per cui fare la prediction

    lover_limit_request = request.query["lower_limit"] # ricevo il limite minore per il filtro (opzionale)
    upper_limit_request = request.query["upper_limit"]  # ricevo il limite superiore per il filtro (opzionale)

    lower_limit = None
    upper_limit = None

    if lover_limit_request != "":
        lower_limit = int(lover_limit_request)
    if upper_limit_request != "":
        upper_limit = int(upper_limit_request)


    if submitted is None:
        return "no text"

    prediction = model_pipeline.predict_proba([submitted])  # faccio la previsione, ricevendo un numpy ndarray

    prediction = prediction[0]
```

```
    prediction_list = prediction.tolist()  # converto l'ndarray interno in una lista

    prediction_float = []  # ricever  i valori della prediction convertiti in float
    for pred in prediction_list:
        pred = float(pred)*100
        pred = round(pred, 2)
        prediction_float.append(pred)

    if lower_limit is not None or upper_limit is not None:  # se uno o pi  dei limiti sono specificati
        returned = run_csp(prediction_float,  # effettuo la ricerca dei valori entro i limiti inseriti
                           lower_limit=lower_limit,
                           upper_limit=upper_limit)
        # il risultato di run_csp sar  una tupla composta da: variabili di csp e risultati della computazione
        results: dict = returned[1]  # prendo i risultati
        variables: set = returned[0]  # prendo le variabili del csp, per interrogare i risultati dopo

        to_return: dict = {}  # dizionario che conterr  solo i risultati entro i limiti specificati
        while len(variables) != 0:  # fino a che ci sono variabili
            var_v = variables.pop()  # prendo il valore
            var_x = variables.pop()  # e il numero x associato al valore, esso determina se il valore    da scartare o no
            if results[var_x] == 1:  # se il numero x    1, non va scartato
                number = int(str(var_x)[1])  # prendo il numero
                to_return[number] = results[var_v]  # lo aggiungo al dizionario da ritornare
    else:  # nel caso in cui non specifico limiti
        to_return: dict = list_to_dict(prediction_float)  # converto la lista della prediction in un dizionario

    response.headers['Access-Control-Allow-Origin'] = '*'
    response.headers['Access-Control-Allow-Methods'] = 'PUT, GET, POST, DELETE, OPTIONS'
    response.headers['Access-Control-Allow-Headers'] = 'Origin, Accept, Content-Type, X-Requested-With, X-CSRF-Token'
    # il codice qui sopra serve per permettere la comunicazione tra interfaccia e server

    return jsondump(to_return)  # ritorno il json dei risultati all'interfaccia


################### i metodi dichiarati sotto servono per debug
@get("/predict_form")
def predict_get():
    global model_pipeline
    if model_pipeline is None:
        return "model not loaded"

    return '''
```

```
        <form action="/predict_form" method="post">
            input: <input name="prediction" type="text" />
            <input value="Predict" type="submit" />
        </form>
    '''


@post("/predict_form")
def predict_post():
    global model_pipeline

    submitted: FormsDict = request.forms.get('prediction')  # se
    mandato via forms

    prediction = model_pipeline.predict_proba([submitted])

    prediction = prediction[0]

    prediction_list = prediction.tolist()
    prediction_integer = []
    for pred in prediction_list:
        pred = float(pred)
        pred *= 100
        pred = round(pred, 2)
        prediction_integer.append(pred)
    returned = run_csp(prediction_integer,
                       lower_limit=None, upper_limit=None)
    results: dict = returned[1]
    variables: set = returned[0]
    to_return: dict = {}
    while len(variables) != 0:
        var_v = variables.pop()
        var_x = variables.pop()
        if results[var_x] == 1:
            number = int(str(var_x)[1])
            to_return[number] = results[var_v]

    return jsondump(to_return)
```

5.2 Client and user GUI

Il client fa richieste al server tramite Typescript

```
    let link = "http://localhost:8080/predict"+"?text="+input+"&
    lower_limit="+lowerLimit.value+"&upper_limit="+upperLimit.value

    console.log(link)
    let response = await fetch(link, {
        method: "POST"
    })

```

La seguente immagine è l'interfaccia grafica creata:

Inoltre è stata creata una versione "portable" come installer .exe o .msi al cui interno è prensente tutto l'essenziale per far eseguire il server ed AI annessa. Si può trovare la repository al seguente link

6 CSP

Abbiamo usato un Constraint Satisfaction Problem per filtrare i risultati delle previsioni se rientrano nell'intervallo specificato in input, segue il modello del CSP:

$$x_n v_n \leq u$$

$$x_n v_n \geq l$$

$x_n \in \{0, 1\}$
$v_n, u, l \in [0, 100]$
$n \in \mathbb{N}$

dove u, l sono rispettivamente *upper_limit* e *lower_limit*.

Nel nostro caso $\mathbb{N} = \{0, 1, ..., 9\}$ in quanto $\mid \mathbb{N} \mid = 10$ per il numero di topics

7 Conclusione

Pur non avendo apportato ottimizzazioni importanti, l'AI è comunque capace di distinguere diversi tipi di input:

- Domande
- Generici statements
- Articoli giornalistici

tutti con buoni risultati.
Ciò non toglie che il modello ML è aperto a molti miglioramenti che tramite metodi di tuning come una applicazione estesa della *GridSearch* può ottenere; Di seguito vengono elencati i parametri che possono essere ottimizzati:

- ngram_range: tuple di valori che specificano la lunghezza del vocabolario dei token che usa, possono essere estese in bigrammi, trigrammi, e così via
- max_df e min_df: valori che il modello usa per eliminare tokens troppo o troppo poco frequenti, a seconda di quali valori sono impostati
- max_features: lunghezza massima dei token vettorializzati, ogni parola viene trasformata in vettori di lunghezza fissa, un aggiustamento di questo valore potrebbe migliorare le performance.
- C: 1.0 è il valore massimo, ma trovarne uno che sia un buon compromesso sortirebbe buoni risultati.