ΠΡΟΤΖΕΚΤ ΕΞΟΡΥΞΗΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΩΝ ΜΑΘΗΣΗΣ

ΟΜΑΔΑ: ΣΤΑΥΡΟΣ ΜΠΟΥΡΑΣ 1053565, ΑΓΓΕΛΟΣ ΡΑΓΚΟΥΣΗΣ 1053566

ΠΕΡΙΒΑΛΛΟΝ ΥΛΟΠΟΙΗΣΗΣ

Χρησιμοποιήθηκε το πρόγραμμα Spyder, που είναι ένα από τα πιο γνωστά προγράμματα για τις εφαρμογές του data mining με χρήση της python. Η εγκαταστασή του ήταν πολύ εύκολη μέσω του Anaconda Navigator (https://docs.anaconda.com/anaconda/navigator/). Και στις δύο ασκήσεις, χρησιμοποιήθηκαν οι βασικές βιβλιοθήκες numpy (περιέχει συναρτήσεις για την επεξεργασία πινάκων) και pandas (περιέχει δομές και συναρτήσεις διαχείρισης των δεδομένων). Στην πρώτη άσκηση χρησιμοποιήθηκε η βιβλιοθήκη sklearn, από την οποία αντλήσαμε συναρτήσεις βασικών αλγορίθμων του data mining που θα αναλυθούν στη συνέχεια (SVC, K-means, Logistic Regression) και άλλες βασικές συναρτήσεις για την ταξινόμηση στοιχείων (π.χ. την train_test_split() για τον διαχωρισμό των δεδομένων σε training και test set, και την classification_report() για την αξιολόγηση του μοντέλου). Στη δεύτερη άσκηση χρησιμοποιήθηκαν επιπρόσθετα κάποιες συναρτήσεις από τη βιβιοθήκη nltk για την επεξεργασία φυσικής γλώσσας που θα αναλυθούν στη συνέχεια (π.χ. πακέτα PorterStemmer και stopwords). Οι περισσότερες βιβλιοθήκες εγκαταστάθηκαν εύκολα μέσω της χρήσης της συνάρτησης download().

ΕΡΩΤΗΜΑ 1

Σας δίνετε το αρχείο winequality-red.csv στο οποίο περιέχονται οίνοι και μετρήσεις που τους χαρακτηρίζουν. Ακόμα περιέχεται και μια εκτίμηση της ποιότητάς τους από κάποιον γευσιγνώστη την οποία και θα πρέπει να μαντέψετε χρησιμοποιώντας την οικογένεια αλγόριθμων κατηγοριοποίησης SVM (Support Vector Machines).

Α. Χωρίστε το dataset σε training-test με αναλογία 75%-25% και να μετρήστε την απόδοσή του μοντέλου σας χρησιμοποιώντας τις μετρικές f1 score, precision και recall. Προσπαθήστε να βελτιώσετε τα αποτελέσματά σας πειραματιζόμενοι με τις παραμέτρους εισόδου.

Β. Σε αυτό το ερώτημα σας ζητείται να αφαιρέσετε το 33% των τιμών του της στήλης pH του training dataset και να προσπαθήσετε να χειριστείτε τις ελλιπείς τιμές με τις ακόλουθες μεθόδους:

1. **Αφαιρέστε τη στήλη**
2. **Συμπληρώστε τις τιμές με το μέσο όρο των στοιχείων της στήλης**
3. **Συμπληρώστε τις τιμές χρησιμοποιώντας Logistic Regression**
4. **Εφαρμόστε K-means και συμπληρώστε τις τιμές που λείπουν με τον αριθμητικό μέσο όρο της συστάδας στην οποία ανήκει το δείγμα.**

Στα νέα μητρώα που προκύπτουν εκπαιδεύστε ένα SVM με τις καλύτερες παραμέτρους που βρήκατε στο υποερώτημα Α και παραθέστε τα ευρήματα σας σχετικά με το πόσο επηρεάστηκε η ποιότητα της κατηγοριοποίησης.

Α) Αρχικά, διαβάζουμε τα δεδομένα του αρχείου «winequality-red.csv» και τα αποθηκεύουμε σε μια μεταβλητή «data» τύπου DataFrame. Διαβάζοντας τη μεταβλητή μέσω του εργαλείου «Variable Explorer» του Spyder, όπως φαίνεται στην εικόνα, βλέπουμε ότι κάθε κρασί αξιολογείται με 12 μετρικές (12 στήλες του πίνακα), όπου

η δωδέκατη στήλη αποτελεί τη ποιότητα του κρασιού. Χωρίζουμε λοιπόν τα δεδομένα σε δύο DataFrames, το «x» που περιλαμβάνει όλες μετρήσεις (11 στήλες) και το «y» που περιλαμάνει μόνο τη στήλη της ποιότητας και άρα τον πίνακα κατηγοριών στις οποίες θα ταξινομηθούν τα κρασιά. Βλέπουμε ότι η ποιότητα κυμαίνεται από 3 έως 8, επομένως υπάρχουν 6 διαθέσιμες κατηγορίες ταξινόμησης. Στη συνέχεια, έγινε τυχαίος διαμοιρασμός του πίνακα x σε 75% training set και 25% test set με τη συνάρτηση train_test_split(), δημιουργώντας τους πίνακες x_tr, y_tr για το training set και x_ts, y_ts για το test set. Έπειτα, δημιουργήσαμε ένα μοντέλο SVC(), μεταβάλλοντας τις τρεις βασικές παραμέτρους του (C, kernel και gamma) και εξετάζοντας ποιές τιμές δίνουν καλύτερα αποτελέσματα. Με τις συναρτήσεις fit() και predict() περάσαμε το training set στο μοντέλο και λάβαμε τον εκτιμώμενο πίνακα κατηγοριών y_pred για το διάνυσμα εισόδου x_tr του test set. Τέλος, με την συνάρτηση classification_report(y_ts, y_pred) συγκρινονται οι εκτιμώμενες με τις πραγματικές τιμές κατηγοριών και παρουσιάζονται τα αποτελέσματα με βάση όλες τις μετρικές (accuracy, f1, precision, recall). Εν τέλη αποδείχτηκε ότι οι τιμές C=100, kernel=rbf, και gamma=auto είχαν τα καλύτερα δυνατά αποτελέσματα.

```
# In[1]: Import Libraries

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report

# In[2]: Import Data

data = pd.read_csv("winequality-red.csv")
x = data.drop('quality', axis='columns')
y = data.quality

# In[3]: Train SVC

x_tr, x_ts, y_tr, y_ts = train_test_split(x,y,test_size=0.25)

model = SVC(C= 100, gamma= 'auto', kernel= 'rbf');
model.fit(x_tr, y_tr)
y_pred=model.predict(x_ts)
print(classification_report(y_ts, y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        11
           5       0.70      0.74      0.72       183
           6       0.62      0.61      0.61       151
           7       0.51      0.48      0.49        48
           8       0.00      0.00      0.00         5

    accuracy                           0.63       400
   macro avg       0.31      0.31      0.31       400
weighted avg       0.62      0.63      0.62       400
```

data - DataFrame

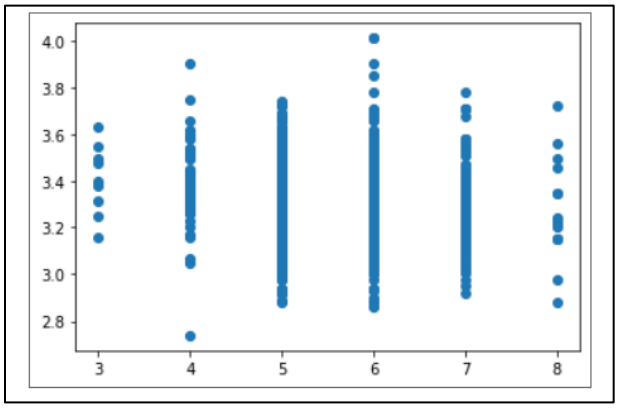
Index	xed acidit	latile acid	citric acid	sidual sug	chlorides	sulfur dio	sulfur dic	density	pH	sulphates	alcohol	quality
0	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	9.8	6
4	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
5	7.4	0.66	0	1.8	0.075	13	40	0.9978	3.51	0.56	9.4	5
6	7.9	0.6	0.06	1.6	0.069	15	59	0.9964	3.3	0.46	9.4	5
7	7.3	0.65	0	1.2	0.065	15	21	0.9946	3.39	0.47	10	7
8	7.8	0.58	0.02	2	0.073	9	18	0.9968	3.36	0.57	9.5	7
9	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
10	6.7	0.58	0.08	1.8	0.097	15	65	0.9959	3.28	0.54	9.2	5
11	7.5	0.5	0.36	6.1	0.071	17	102	0.9978	3.35	0.8	10.5	5
12	5.6	0.615	0	1.6	0.089	16	59	0.9943	3.58	0.52	9.9	5
13	7.8	0.61	0.29	1.6	0.114	9	29	0.9974	3.26	1.56	9.1	5
14	8.9	0.62	0.18	3.8	0.176	52	145	0.9986	3.16	0.88	9.2	5
15	8.9	0.62	0.19	3.9	0.17	51	148	0.9986	3.17	0.93	9.2	5

B1] Σε αυτό το ερώτημα ζητείται να εκπαιδεύσουμε και να αξιολογήσουμε πάλι το μοντέλο αφαιρώντας τελειώς τη στήλη του pH. Αυτό υλοποιείται εύκολα μέσω της συνάρτησης drop() για τον πίνακα εισόδου data, όπου εκτός από τη στήλη «quality» αφαιρούμε πλέον και τη στήλη του «pH». Ο υπόλοιπος κώδικας είναι ίδιος. Παρατηρώντας πάλι τα αποτελέσματα, βλέπουμε ότι οι μεταβολές των μετρικών είναι πάρα πολύ μικρές αγνοώντας τελειώς την επίδραση της μετρικής του pH. Συνεπώς συμπεραίνουμε ότι οι μετρήσεις του pH δεν επηρεάζουν σημαντικά την ταξινόμηση.

```
# In[3]: Erotima B1

data = pd.read_csv("winequality-red.csv")
x = data.drop(['quality', 'pH'], axis='columns')
y = data.quality

x_tr, x_ts, y_tr, y_ts = train_test_split(x,y,test_size=0.25)

model = SVC(C= 100, gamma= 'auto', kernel= 'rbf');
model.fit(x_tr, y_tr)
y_pred=model.predict(x_ts)
print(classification_report(y_ts, y_pred))
```

	precision	recall	f1-score	support
3	0.00	0.00	0.00	4
4	0.00	0.00	0.00	11
5	0.71	0.70	0.70	178
6	0.52	0.54	0.53	153
7	0.50	0.47	0.48	51
8	0.00	0.00	0.00	3
accuracy			0.58	400
macro avg	0.29	0.29	0.29	400
weighted avg	0.58	0.58	0.58	400

Επιχειρώντας να ερμηνεύσουμε τη μικρή επίδραση της στήλης του pH στην ταξινόμηση, σχεδιάζουμε τη γραφική παράσταση των τιμών του pH (κάθετη στήλη) ως προς τις τιμές των 6 κατηγοριών της στήλης quality. Παράλληλα, μετράμε πόσα κρασιά υπάρχουν ανά κατηγορία. Βλέπουμε ότι η συντριπτική πλειοψηφία των κρασιών ανήκουν στις κατηγορίες 5,6 και 7, και από τη γραφική παρατηρούμε ότι τα κρασιά και στις 3 αυτές κατηγορίες έχουν τα περισσότερα τιμή pH περίπου μέσα στο πολύ μικρό διάστημα 3.0 έως 3.6. Επειδή λοιπόν και οι τρεις κυρίαρχες κατηγορίες μοιράζονται το ίδιο μικρό διάστημα και όχι διακριτές περιοχές, οι κατηγορίες θα έχουν παρόμοια επίδραση στη ταξινόμηση οποιουδήποτε νέου παραδείγματος με βάση το pH, επομένως συνολικά οι μετρήσεις του pH δεν βοηθάνε αισθητά στην ταξινόμηση.



5	681
6	638
7	199
4	53
8	18
3	10

```
matplotlib.pyplot.scatter(data.quality,data.pH)
matplotlib.pyplot.show()
print(data['quality'].value_counts())
```

B2] Στο ερώτημα αυτό ζητείται να αφαιρεθεί το 33% των τιμών της στήλης του pH και στη συνέχεια να συμπληρωθούν οι τιμές αυτές με το μέσο όρο των υπολοίπων τιμών της στήλης. Προσθέσαμε λοιπόν το παρακάτω τμήμα κώδικα, το οποίο αρχικά υπολογίζει το μέσο όρο mean διαιρώντας το άθροισμα όλων των στοιχείων της στήλης με το πλήθος τους, και στη συνέχεια επιλέγει τυχαία 396 τιμές της στήλης (33%) και τις αντικαθιστά με το μέσο όρο. Εφόσον το διάστημα των τιμών του pH είναι μικρό, η συμπλήρωση με την τιμή του μέσου όρου είναι μια καλή προσέγγιση των τιμών που αγνοούνται, όμως και πάλι βλέποντας τα αποτελέσματα βλέπουμε μικρές αλλαγές εφόσον γενικότερα το pH έχει μικρή επίδραση στην ταξινόμηση για τους λόγους που αναφέρθηκαν.

```
sum = 0
for k in x_tr.pH:
    sum = sum + k
mean = sum/x_tr.pH.count()

for r in range (0,396):
    i  = random.randint(0,1200)
    while (i not in x_tr.pH.index or x_tr.pH[i] == mean):
        i  = random.randint(0,1200)
    x_tr.pH[i] = mean
```

	precision	recall	f1-score	support
3	0.50	0.20	0.29	5
4	0.10	0.07	0.08	14
5	0.64	0.66	0.65	172
6	0.61	0.61	0.61	158
7	0.52	0.53	0.53	49
8	0.00	0.00	0.00	2
accuracy			0.59	400
macro avg	0.39	0.35	0.36	400
weighted avg	0.59	0.59	0.59	400

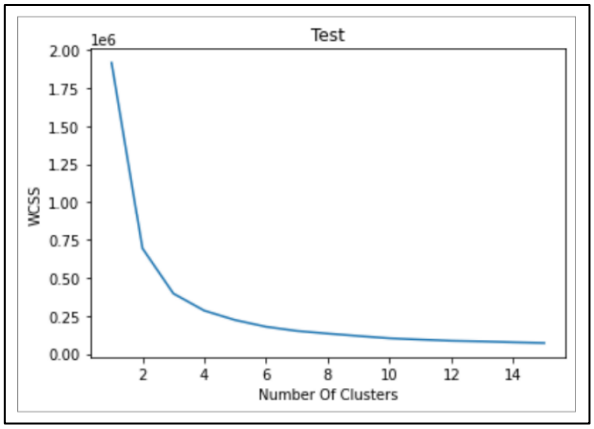
Β3] Για τη συμπλήρωση των τιμών που λείπουν με τη μέθοδο του Logistic Regression, αρχικά χωρίζουμε όπως πάντα τα δεδομένα σε ένα training set 75% (x_tr) και ένα test set 25% (x_ts), και επιλέγουμε το 33% των τιμών του training set να έχουν ελλειπή τιμή pH. Στη συνέχεια, εφόσον θέλουμε να προβλέψουμε τις τιμές pH που λείπουν στο training set, το χωρίζουμε περαιτέρω σε ένα training set με όλα τα στοιχεία του που έχουν τιμή pH (x_tr_with_pH) και σε ένα test set με όλα τα στοιχεία του που δεν έχουν τιμή pH (x_tr_without_pH). Δημιουργώντας λοιπόν ένα μοντέλο ταξινομητή με τη συνάρτηση LogisticRegression(), δημιουργείται ένας πίνακας y_pH_pred με τις τιμές pH που προβλέφθηκαν για τα στοιχεία του training set που δεν έχουν pH, και στη συνέχεια με ένα for loop προσθέτουμε τις τιμές αυτές στον αρχικό πίνανα του training set (x_tr). Ύστερα, εκτελούμε την ίδια διαδικασία με τα υπόλοιπα ερωτήματα (ορίζουμε τον SVC ταξινομητή κτλπ), και παρατηρούμε πάλι ότι τα αποτελέσματα κυμαίνονται στα ίδια επίπεδα, για τους λόγους που έχουμε ήδη αναφέρει πιο πάνω.

```
x_tr, x_ts, y_tr, y_ts = train_test_split(x,y,test_size=0.25, random_state=2)
x_tr.loc[x_tr.sample(frac=0.33,random_state=4).index,'pH'] = np.nan

label_encoder = preprocessing.LabelEncoder()
samples = x_tr.sample(frac=0.33,random_state=4).index

y_tr_with_pH = label_encoder.fit_transform((data[~data.index.isin(samples)])['pH'])
x_tr_with_pH = (data[~data.index.isin(samples)]).drop('pH', axis=1)
y_pH_pred = (data[data.index.isin(samples)])['pH']
x_tr_without_pH = (data[data.index.isin(samples)]).drop('pH', axis=1)

model = LogisticRegression()
model.fit(x_tr_with_pH, y_tr_with_pH)
y_pH_pred = model.predict(x_tr_without_pH)
y_pH_pred = label_encoder.inverse_transform(y_pH_pred)
cnt = 0;

for index in x_tr_without_pH.index:
    (x_tr.loc[index])['pH']=y_pH_pred[cnt]
    cnt=cnt+1
```

	precision	recall	f1-score	support
3	0.00	0.00	0.00	1
4	0.25	0.08	0.12	13
5	0.56	0.74	0.64	148
6	0.66	0.52	0.58	185
7	0.43	0.48	0.46	48
8	1.00	0.20	0.33	5
accuracy			0.58	400
macro avg	0.48	0.34	0.35	400
weighted avg	0.59	0.58	0.57	400

Β4] Για τη συμπλήρωση των τιμών που λείπουν με τη μέθοδο του K-means, παίρνουμε το σύνολο των δεδομένων και τα ομαδοποιούμε σε clusters με βάση όλα τα γνωρίσματα τους εκτός από το pH. Στη συνέχεια, υπολογίζουμε σε κάθε cluster το μέσο όρο του pH των στοιχείων του που έχουν έγκυρη τιμή, και συμπληρώνουμε τις τιμές pH των υπόλοιπων στοιχείων του cluster που δεν έχουν τιμή με την τιμή του μέσου όρου αυτού. Αρχικά λοιπόν, κατασκευάζουμε το διάγραμμα elbow, δοκιμάζοντας τον αλγόριθμο k-Means για 1 έως 16 clusters, με στόχο να προσδιορίσουμε τον βέλτιστο αριθμό από clusters. Όπως παρατηρούμε από τη γραφική, ο βέλτιστος αριθμός clusters είναι περίπου 4.

```
for i in range(1,16):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=0)
    kmeans.fit(feature_df)
    wcss.append(kmeans.inertia_)

matplotlib.pyplot.plot(range(1,16),wcss)
matplotlib.pyplot.title('Test')
matplotlib.pyplot.xlabel('Number Of Clusters')
matplotlib.pyplot.ylabel('WCSS')
matplotlib.pyplot.show()
```



Στη συνέχεια, εφαρμόζουμε τον αλγόριθμο K-means στα δεδομένα μας για 4 clusters, αφού πρώτα έχουμε αφαιρέσει όλη τη στήλη του pH για να μην ληφθεί υπόψη στην ομαδοποίηση. Η συνάρτηση kmeans.fit_predict(feature_df) μας επιστρέφει τα δεδομένα σε 4 clusters, τα οποία μετά τα αναθέτουμε σε 4 ξεχωριστές λίστες k1,k2,k3,k4. Έπειτα, σε κάθε k εισάγουμε τις τιμές pH των στοιχείων του που έχουν pH από τον αρχικό πίνακα δεδομένων data με την συνάρτηση insert(), και σε όσα δεν έχουν pH εισάγουμε το μέσο όρο pH του cluster μέσω της συνάρτησης fillna(). Τέλος, με την συνάρτηση concat() ενώνουμε πάλι τα clusters σε μια ενιαία λίστα και εκτελούμε ύστερα την ίδια διαδικασία με τα υπόλοιπα ερωτήματα (ορίζουμε τον SVC ταξινομητή κτλπ). Και πάλι βλέπουμε ότι τα αποτελέσματα κυμαίνονται στα ίδια επίπεδα, για τους λόγους που έχουμε ήδη αναφέρει πιο πάνω.

```
kmeans=KMeans(n_clusters=4, init='k-means++',random_state=0)
y=kmeans.fit_predict(feature_df)

feature_df['Clusters']=y
k1=feature_df[y==0]
k1.insert(8,'pH',data['pH'])
k2=feature_df[y==1]
k2.insert(8,'pH',data['pH'])
k3=feature_df[y==2]
k3.insert(8,'pH',data['pH'])
k4=feature_df[y==3]
k4.insert(8,'pH',data['pH'])

k1.fillna(k1.mean(), inplace=True)
k2.fillna(k2.mean(), inplace=True)
k3.fillna(k3.mean(), inplace=True)
k4.fillna(k4.mean(), inplace=True)

new_celldf1=pd.concat([k1,k2,k3,k4],ignore_index=False)
```

	precision	recall	f1-score	support
3	0.00	0.00	0.00	1
4	0.00	0.00	0.00	13
5	0.61	0.72	0.66	159
6	0.60	0.53	0.57	172
7	0.44	0.47	0.46	49
8	0.00	0.00	0.00	6
accuracy			0.57	400
macro avg	0.28	0.29	0.28	400
weighted avg	0.56	0.57	0.56	400

ΕΡΩΤΗΜΑ 2

Σε αυτό το ερώτημα σας δίνεται το αρχείο onion-or-not.csv το οποίο περιέχει δύο στήλες. Η πρώτη στήλη περιλαμβάνει τίτλους από ψευδείς ειδήσεις ενώ η δεύτερη στήλη μας πληροφορεί αν αυτές δημοσιεύθηκαν στο γνωστό χιουμοριστικό website theonion.com ή όχι. Σκοπός σας είναι να προσπαθήσετε να μαντέψετε την πληροφορία της δεύτερης στήλης χρησιμοποιώντας ένα νευρωνικό δίκτυο. Για να μετασχηματίσετε τους τίτλους των ταινιών έτσι ώστε να

δημιουργήσετε το μητρώο το οποίο θα δώσετε ως είσοδο στο υπό εκπαίδευση μοντέλο θα πρέπει να ακολουθήσετε την παρακάτω διαδικασία:

1. **Θα χωρίσετε τους τίτλους σε λέξεις, δημιουργώντας ένα διάνυσμα λέξεων.**
2. **Από τις λέξεις θα αφαιρέσετε τις καταλήξεις τους, κρατώντας μόνο το θέμα τους (stemming).**
3. **Θα αφαιρέσετε από την συλλογή σας εκείνες τις λέξεις που είναι αρκετά κοινές και δεν προσφέρουν πληροφορία (stopwords removal).**
4. **Στις εναπομείνασες λέξεις θα αναθέσετε ως βάρος την τιμή tf-idf.**
5. **Θα συνδυάσετε τα διανύσματα σας για να παραχθεί το τελικό μητρώο.**

Μετά τη δημιουργία του μητρώου, καλείστε να το χωρίσετε σε training-test dataset με αναλογία 75%-25%. Στη συνέχεια, θα πρέπει να εκπαιδεύσετε ένα νευρωνικό δίκτυο (όποιου τύπου επιθυμείτε εσείς) και να μετρήσετε την απόδοσή του χρησιμοποιώντας τις μετρικές f1 score, precision και recall.

Υπόδειξη

Για τις ενέργειες επεξεργασίας φυσικής γλώσσας, μπορείτε να χρησιμοποιήσετε το εργαλείο της python με όνομα NLTK (Natural Language Toolkit)

Α] ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ

Αρχικά εισάγουμε τα δεδομένα του αρχείου «onion-or-not.csv» σε μια μεταβλητή «df» τύπου DataFrame. Διαβάζοντας τη μεταβλητή μέσω του εργαλείου «Variable Explorer» του Spyder, όπως φαίνεται στην εικόνα, βλέπουμε ότι τα δεδομένα περιλαμβάνουν 24.000 γραμμές και δύο στήλες, όπου η πρώτη στήλη «text» περιλαμβάνει μία πρόταση για κάθε γραμμή και η δεύτερη στήλη «label» δηλώνει την κατηγορία στην οποία ανήκει, δηλαδή εάν η πρόταση αυτή δημοσιεύθηκε στο γνωστό χιουμοριστικό website theonion.com ή όχι. Χωρίζουμε λοιπόν το DataFrame σε δύο επιμέρους DataFrames «feature_cell» και «yes_or_no», όπου το πρώτο περιέχει μόνο τη στήλη των 24.000 προτάσεων και το δεύτερο μόνο τη στήλη της κατηγορίας στην οποία ανήκουν.

Index	text	label
0	Entire Facebook Staff Laughs As Man Tightens Privacy Settings	1
1	Muslim Woman Denied Soda Can for Fear She Could Use It as a Weapon	0
2	Bold Move: Hulu Has Announced That They're Gonna Go Ahead And Reboot 'Shrill' While It's Still On Since You Idiots Will Watch Anything	1
3	Despondent Jeff Bezos Realizes He'll Have To Work For 9 Seconds To Earn Back Money He Lost In Divorce	1
4	For men looking for great single women, online dating offers a viable solution to the otherwise frustrating task of finding long-term love. A0rNsK5	1
5	Kim Jong-Un Wonders If Nuclear Threats Distracting Him From Real Goal Of Starving Citizenry	1
6	Omaha dad finds pot brownies, eats 4 of them, says mean things to cat	0
7	Pokémon Go player stabbed, keeps playing	0
8	Job Placement Service Helps Students Who Fail Out Of Dad's Alma Mater Find Work At Dad's Company	1
9	Idiot Zoo Animal With Zero Predators Still Protective Of Young	1
10	Point/Counterpoint: Oh, Are The PC Police Here To Arrest Me For Havin' Opinions? vs. Sir, We Are The Regular Police And You Need To Come Out Of That Slide	1
11	Woman's rejected "8theist" license plate violates First Amendment	0
12	Man Tries to Rob a Bank After Paying $500 to a Wizard to Make Him Invisible	0
13	Hubble Telescope Desperately Struggling To Contact NASA After Witnessing Murder On Ganymede	1
14	Incredibly Sad: This Guy Got A New Blender And It Literally Changed His Life	1
15	Ohio parents sue their olympic gold medalist daughter for telling stories that cast them in a bad light... so everyone knows that they are not selfish, don't bully her and ha…	0
16	Match-predicting koala sacked for poor results	0
17	Nation Demands New Photograph Of Edward Snowden	1
18	Gaunt, Hollow-Eyed Big Bird Enters Sixth Day Of Hunger Strike Against Proposed Trump Budget	1
19	Court: 93-Year-Old Kills Wife Of 70 Years Because 'He Couldn't Take It Anymore'	0
20	What's Next For Hillary Clinton?	1
21	Life: Animal Rights FTW! Popeyes Has Announced That They Will Only Serve Chickens Killed In Self-Defense!	1
22	Man Putting Off Starting Family To Focus On Treading Water In Career For Few Years	1
23	Coach's Un-Athletic Son Going To Get Fucking Reamed After Game For Dropping Fly Ball	1
24	Deputies ask man to quit calling about his stolen marijuana	0

```
df = pd.read_csv("onion-or-not.csv")

feature_cell=df[['text']]

yes_or_no=df[['label']] ##input for neural net
```

Στη συνέχεια, κάνουμε την προεπεξεργασία των προτάσεων εισόδου. Αρχικά, μέσω της συνάρτησης nltk.word_tokenize() εισάγουμε διαδοχικά κάθε μία από τις προτάσεις δεδομένων με ένα for loop και τις χωρίζουμε σε μεμονωμένες λέξεις. Η συνάρτηση επιστρέφει μια λίστα με τις μεμονωμένες λέξεις κάθε γραμμής, και έτσι προσθέτουμε κάθε λίστα-γραμμή σε μια ενιαία λίστα «all_tokenized» που περιέχει όλες τις λέξεις του αρχείου (η τελική λίστα είναι επομένως μια διπλή λίστα). Στη συνέχεια, εισάγουμε ένα στιγμιότυπο «ps» τύπου PorterStemmer της βιβλιοθήκης nltk, από το οποίο καλούμε τη συνάρτηση stem() για κάθε λέξη της διπλής λίστας και αποθηκεύουμε τις νέες λέξεις σε μια νέα διπλή λίστα «all_stemmed». Η διαδικασία του stemming είναι απαραίτητη διότι μειώνει σημαντικά το πλήθος των διαφορετικών ομόρριζων λέξεων, κρατώντας μόνο το θέμα τους και αφαιρώντας την κατάληξή τους.

```
ps=PorterStemmer()

all_tokenized=[]
all_stemmed=[]

for index, row in df.iterrows():
    all_tokenized.append(nltk.word_tokenize(row['text']))

for i in all_tokenized:
    current_stemmed=[]
    for x in i:
        current_stemmed.append(ps.stem(x))
    all_stemmed.append(current_stemmed)
```

Έπειτα, διαγράφουμε από τη λίστα των λέξεών μας όσες λέξεις είναι περιττές για την εκπαίδευση του μοντέλου μας. Οι λέξεις αυτές της αγγλικής γλώσσας περιλαμβάνονται σε έναν πίνακα «stopwords» της βιβλιοθήκης nltk. Επομένως, εξετάζουμε πάλι κάθε λέξη της λίστας μας και βλέπουμε εάν περιλαμβάνεται στον πίνακα stopwords και άρα εάν πρέπει να παραληφθεί. Όσες λέξεις δεν περιλαμβάνονται, τις αποθηκεύουμε σε μια νέα λίστα δεδομένων «all_filtered».

```
all_filtered=[]

for i in all_stemmed:
    stop_words = set(stopwords.words('english'))
    current_filtered = []
    for w in i:
        if w not in stop_words:
            current_filtered.append(w)

    all_filtered.append(current_filtered)

del all_stemmed,all_tokenized
```

Στη συνέχεια, για να εφαρμόσουμε την τεχνική tf-idf, δημιουργούμε αρχικά μια μεταβλητή «uniqueWords» τύπου Set, στην οποία τοποθετούμε μία μόνο φορά κάθε ξεχωριστή λέξη της λίστας μας, παραλείποντας έτσι τις επαναλήψεις ιδίων λέξεων. Βλέπουμε ότι οι διαφορετικές λέξεις του κειμένου μας είναι 20.456, μέσα στις 24.000 προτάσεις. Στη συνέχεια, για κάθε μια από τις προτάσεις μας δημιουργούμε ένα λεξικό «my_dict», το οποίο έχει ως κλειδιά όλες τις 20.456 ξεχωριστές λέξεις του κειμένου και στο οποίο βάζουμε ως τιμή σε κάθε λέξη τον αριθμό των επαναλήψεων της μέσα στη πρόταση. Εάν για παράδειγμα υπήρχε η πρόταση «Bob eats pizza», θα δίναμε την τιμή 1 στα κλειδιά «Bob», «eats» και «pizza», και τη τιμή 0 στα υπόλοιπα 20.453 κλειδιά

του λεξικού της πρότασης. Φτιάχουμε έτσι συνολικά 24.000 λεξικά των 20.456 κλειδιών-λέξεων, και τα εισάγουμε στη νέα μας λίστα «all_dictionaries». Καταφέρνουμε έτσι να κάνουμε όλες τις προτάσεις να έχουν το ίδιο μήκος (λεξικά των 20.456 κλειδιών) και να φτιάξουμε επομένως μια ομοιόμορφη λίστα.

```
uniqueWords=set()

for i in all_filtered:
    uniqueWords = uniqueWords.union(set(i))

all_dictionaries=[]

for i in all_filtered:

    my_dict= dict.fromkeys(uniqueWords, 0)

    for word in i:
        my_dict[word] += 1

    all_dictionaries.append(my_dict)
```

Υπολογίζουμε έπειτα την IDF κάθε λέξης (Inverse document frequency), η οποία εκφράζει πρακτικά σε πόσες προτάσεις του κειμένου εμφανίζεται η συγκεκριμένη λέξη. Η συνάρτηση computeIDF() που δημιουργήσαμε παίρνει ως όρισμα τη λίστα των λεξικών και επιστρέφει ένα λεξικό «idf-list» που έχει ως κλειδιά τις 20.456 ξεχωριστές λέξεις και ως τιμές τις IDF των λέξεων.

```
idf_list=[]

idf_list=computeIDF(all_dictionaries)
```

```
def computeIDF(documents):
    import math
    N = len(documents)

    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1

    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict
```

Υπολογίζουμε έπειτα την TF κάθε λέξης (Term Frequency), η οποία εκφράζει πρακτικά πόσες φορές εμφανίζεται η συγκεκριμένη λέξη σε κάθε πρόταση. Η συνάρτηση computeTF() που δημιουργήσαμε καλείται 24.000 (μία φορά για κάθε πρόταση) και επιστρέφει ένα λεξικό tf-list για κάθε πρόταση που έχει ως κλειδιά τις 20.456 ξεχωριστές λέξεις και ως τιμές τις TF των λέξεων της πρότασης. Πολλαπλασιάζοντας τέλος κάθε λίστα «tf-list» με την λίστα «idf-list» μέσω της συνάρτησης computeTFIDF(), δημιουργούμε ένα λεξικό «tf_idf_list» για κάθε πρόταση, και βάζουμε τα συνολικά 24.000 λεξικά που δημιουργούνται σε μια τελική διπλή λίστα «final_array», αφαιρώντας τα ονόματα των λέξεων και κρατώντας μόνο την TF-IDF τιμή κάθε λέξης. Η τελική λίστα έχει διαστάσεις 24.000 (γραμμές) x 20.456 (TF-IDF τιμές των λέξεων).

```
tf_idf_list=[]
for i in range(24000):
    tf_list= ((computeTF(all_dictionaries[i],all_filtered[i])))
    tf_idf_list.append(computeTFIDF(tf_list,idf_list))

del all_dictionaries,tf_list


final_array=[]

for d in tf_idf_list:
    temp=[]
    temp=list(d.values())
    final_array.append(temp) ##input for neural net
```

```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf
```

```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
```

Β] ΕΚΠΑΙΔΕΥΣΗ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ

Για την εκπαίδευση του νευρωνικού δικτύου, μετασχηματίζουμε αρχικά την τελική μας λίστα «final_array» ως πίνακα X, και την λίστα των κατηγοριών «yes_or_no» ως πίνακα Y. Χωρίζουμε τον πίνακα X σε 75% training set και 25% test set, και εκπαιδεύουμε ένα νευρωνικό δίκτυο με τις παρακάτω προδιαγραφές.

```
X=np.asarray(final_array,dtype=np.float16)
y=np.asarray(yes_or_no,dtype=np.int16)
```

```
X_train, X_test, y_train, y_test= train_test_split(X,y,test_size= 0.25,random_state=4)

# # define the keras model
model = Sequential()
model.add(Dense(18000, input_dim=20456, activation='relu',kernel_initializer='he_uniform'))
model.add(Dense(20456, activation='relu'))
model.add(Dense(2000, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[f1_m,precision_m, recall_m])

history = model.fit(X_train, y_train, epochs=4)

# # evaluate the model
loss,f1_score, precision, recall = model.evaluate(X_test, y_test, verbose=0,batch_size=32)
```

Γ] ΑΠΟΤΕΛΕΣΜΑΤΑ

Παρατηρώντας τα αποτελέσματα, βλέπουμε ότι το πολυεπίπεδο νευρωνικό δίκτυο έχει πολύ ικανοποιητικές επιδόσεις, κάνοντας αξιολόγηση με το test set. Συγκεκριμένα, προκύπτει ότι **precision = 0.8837**, **f1_score = 0.8510**, και **recall = 0.8306**.

f1_score	float	1	0.8510547280311584
history	callbacks.callbacks.History	1	History object of keras.callbacks.callbacks module
loss	float	1	0.8139640821845581
precision	float	1	0.8837457299232483
recall	float	1	0.8306763768196106
stopwords	corpus.util.LazyCorpusLoader	1	LazyCorpusLoader object of nltk.corpus.util module