```
! pip install levenshtein
! pip install jarowinkler
! pip install py_stringmatching
! pip install pyjarowinkler
! pip install utils
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: levenshtein in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: rapidfuzz<3.0.0,>=2.3.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: jarowinkler<2.0.0,>=1.2.2 in /usr/local/lib/python3.7/
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: jarowinkler in /usr/local/lib/python3.7/dist-packages
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: py_stringmatching in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from py
Requirement already satisfied: numpy>=1.7.0 in /usr/local/lib/python3.7/dist-packages
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: pyjarowinkler in /usr/local/lib/python3.7/dist-package
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Requirement already satisfied: utils in /usr/local/lib/python3.7/dist-packages (1.0.
```

```
import pandas as pd
from google.colab import drive
import pandas as pd
from scipy import stats
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import Levenshtein as Lev
from itertools import product
from py_stringmatching import similarity_measure as sm
from pyjarowinkler import distance
import utils
import seaborn as sns
import itertools
from sklearn.preprocessing import StandardScaler
import time


drive.mount('/content/gdrive')
# !ls "/content/gdrive/My Drive"
data_path = "/content/gdrive/My Drive/Master ADS/Week 3/data/"
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive
```

Task 1

Question 1. The two tables can be merged using the Accident_Index field. Write Python code to merge the two tables and store the results in a new csv file.

```
accidents = pd.read_csv(data_path + "Accidents_2015.csv")
casualties = pd.read_csv(data_path + "Casualties_2015.csv")
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWa
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
# explore the dataset
accidents
```

	Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	L
0	201501BS70001	525130.0	180050.0	-0.198465	51
1	201501BS70002	526530.0	178560.0	-0.178838	51
2	201501BS70004	524610.0	181080.0	-0.205590	51
3	201501BS70005	524420.0	181080.0	-0.208327	51
4	201501BS70008	524630.0	179040.0	-0.206022	51
...	...	...	...	...	
140051	2015984139115	312087.0	570791.0	-3.376671	55
140052	2015984139715	320671.0	569791.0	-3.242159	55
140053	2015984140215	311731.0	586343.0	-3.387067	55
140054	2015984140515	328273.0	570137.0	-3.123385	55
140055	2015984141415	314050.0	579638.0	-3.348646	55

140056 rows × 32 columns

```
casualties
```

	Accident_Index	Vehicle_Reference	Casualty_Reference	Casualty_Class	Sex_o
0	201597UA71710	2	1	1	
1	201597UA71810	2	1	2	
2	201597UA71810	2	2	2	
3	201597UA71810	2	3	2	
4	201597UA71810	2	4	2	

```
merged_df = pd.merge(accidents, casualties, on = "Accident_Index")
```

```
# save to csv
merged_df.to_csv(data_path + "merged_df.csv")
```

Question 2. The Accident_Severity variable needs to be recoded. Write Python code to replace the values in this column as: 1=Minor (1 should be converted to minor), 2=Medium, 3=Severe.

```
severity = [1, 2, 3]
string = ["Minor", "Medium", "Severe"]

merged_df["Accident_Severity"] = merged_df["Accident_Severity"].replace(severity, string)
merged_df
```

	Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	La
0	201501BS70001	525130.0	180050.0	-0.198465	51
1	201501BS70002	526530.0	178560.0	-0.178838	51
2	201501BS70004	524610.0	181080.0	-0.205590	51
3	201501BS70005	524420.0	181080.0	-0.208327	51
4	201501BS70008	524630.0	179040.0	-0.206022	51
...	...	...	...	...	
164515	2015984141415	314050.0	579638.0	-3.348646	55
164516	2015984141415	314050.0	579638.0	-3.348646	55
164517	2015984141415	314050.0	579638.0	-3.348646	55
164518	2015984141415	314050.0	579638.0	-3.348646	55
164519	2015984141415	314050.0	579638.0	-3.348646	55

164520 rows × 47 columns

Question 3. Replace missing values in a set of attributes by -1 (this process has been already done for the example datasets). Then, write Python code to detect these values and report the names of the columns in each table that contain such values.

```
merged_df = merged_df.fillna(-1)
# how many columns have missing values
print(len(merged_df.columns[merged_df.isin([-1]).any()]))
# name of columns that have missing values
print(merged_df.columns[merged_df.isin([-1]).any()])
```

```
26
Index(['Location_Easting_OSGR', 'Location_Northing_OSGR', 'Longitude',
       'Latitude', 'Time', 'Junction_Detail', 'Junction_Control',
       '2nd_Road_Class', '2nd_Road_Number',
       'Pedestrian_Crossing-Human_Control',
       'Pedestrian_Crossing-Physical_Facilities', 'Road_Surface_Conditions',
       'Special_Conditions_at_Site', 'Carriageway_Hazards',
       'Did_Police_Officer_Attend_Scene_of_Accident',
       'LSOA_of_Accident_Location', 'Sex_of_Casualty', 'Age_of_Casualty',
       'Age_Band_of_Casualty', 'Pedestrian_Location', 'Pedestrian_Movement',
       'Car_Passenger', 'Bus_or_Coach_Passenger',
       'Pedestrian_Road_Maintenance_Worker', 'Casualty_Home_Area_Type',
       'Casualty_IMD_Decile'],
      dtype='object')
```

Question 4. For all numerical variables, write a Python function to check if there are any clearly extreme values, or values that do not belong in that column. If you find any, remove these records from the dataset.

```
# get only the numerical values
merged_df_numerical = merged_df.select_dtypes(include = np.number)
# get the rows in which there are no outliers
# for all the columns in the dataset
merged_df_no_outliers = merged_df_numerical[(np.abs(stats.zscore(merged_df_numerical)) < 3
# index the original dataset with the indexes that
# have no outliers to include the non-numerical columns
merged_df_no_outliers = merged_df.loc[merged_df_no_outliers.index]
```

Question 5. Write Python code to create a new attribute (column) called is_minor, that checks whether a casualty was a minor or an adult. Being adult is defined as having an age of 18 or above. The column should only contain the values ‘Yes’ and ‘No’.

```
merged_df_no_outliers['is_minor'] = np.where(merged_df_no_outliers['Age_of_Casualty'] < 18
```

```
merged_df_no_outliers['Age_of_Casualty'].tail(10)
```

```
164491    27
164493    72
164494    61
```

```
    164495    17
    164501    62
    164502    46
    164504    38
    164505    24
    164506    22
    164508    24
    Name: Age_of_Casualty, dtype: int64
```

```
merged_df_no_outliers['is_minor'].tail(10)
```

```
    164491     No
    164493     No
    164494     No
    164495    Yes
    164501     No
    164502     No
    164504     No
    164505     No
    164506     No
    164508     No
    Name: is_minor, dtype: object
```

Question 6. Choose an attribute which is numeric and has some missing values. Then, calculate the average of all the available values in that column and fill the missing cells in the column with the average value. For example, the Location_Easting_OSGR variable has about 27 missing values - solve this with imputation of the average of the 'Location_Easting_OSGR' of all records.

```
merged_df_no_outliers["Location_Easting_OSGR"] = merged_df_no_outliers["Location_Easting_O
```

Task 2

Question 1. Remove the disguised values from the table. We need to remove the values that equal to 0 from columns BloodPressure, SkinThickness and BMI as these are missing values but they have been disguised by the value 0. Remove the value but keep the record (i.e.) change the value to null.

```
# loading the diabetes dataset
diabetes = pd.read_csv(data_path + "diabetes.csv")
y = diabetes["Outcome"]
```

```
# creating a copy to do this question
diabetes_null = diabetes.copy()
```

```
# checking for null values
diabetes_null.isnull().sum()
```

```
    Pregnancies                 0
    Glucose                     0
```

```
     BloodPressure               0
     SkinThickness               0
     Insulin                     0
     BMI                         0
     DiabetesPedigreeFunction    0
     Age                         0
     Outcome                     0
     dtype: int64
```

```
# replacing 0 with NaN values
diabetes_null["BloodPressure"] = diabetes_null["BloodPressure"].replace(0, np.nan)
diabetes_null["SkinThickness"] = diabetes_null["BloodPressure"].replace(0, np.nan)
diabetes_null["BMI"] = diabetes_null["BMI"].replace(0, np.nan)
```

```
# checking if the code bellow worked
diabetes_null.isnull().sum()
```

```
     Pregnancies                  0
     Glucose                      0
     BloodPressure               35
     SkinThickness               35
     Insulin                      0
     BMI                         11
     DiabetesPedigreeFunction     0
     Age                          0
     Outcome                      0
     dtype: int64
```

Question 2. Remove the Outcome column and remove one of the columns if their correlation is greater than 0.5. That is, if there are two columns with correlation value > 0.5 then remove one of them and keep the other. The input for this step is the original dataframe not the one that has been produced at step 1.

```
# drop de outcome column
diabetes = diabetes.drop(columns=["Outcome"])
```

```
# show correlations between attributes
corrMatrix = diabetes.corr().abs()
corrMatrix
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insuli
Pregnancies	1.000000	0.129459	0.141282	0.081672	0.07353
Glucose	0.129459	1.000000	0.152590	0.057328	0.33135

```
# create the upper triangle of the correlation matrix
# it doesnt matter whether we use the lower or the upper
# because they are mirrors but it is important that we dont use the diagonal
upper_triangle = corrMatrix.where(np.triu(np.ones(corrMatrix.shape), k=1).astype(np.bool))
# filtering by a specific correlation
to_drop = [column for column in upper_triangle.columns if any(upper_triangle[column] > 0.5
diabetes.drop(to_drop, axis=1, inplace=True)
```

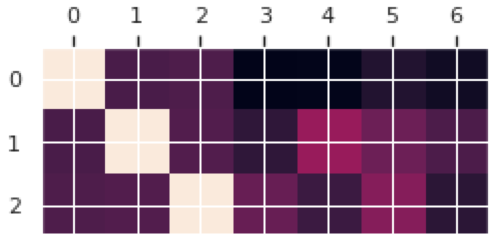
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DeprecationWarning: `
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
  after removing the cwd from sys.path.
```

```
diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedig
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 7 columns

```
plt.matshow(diabetes.corr())
plt.show()
```

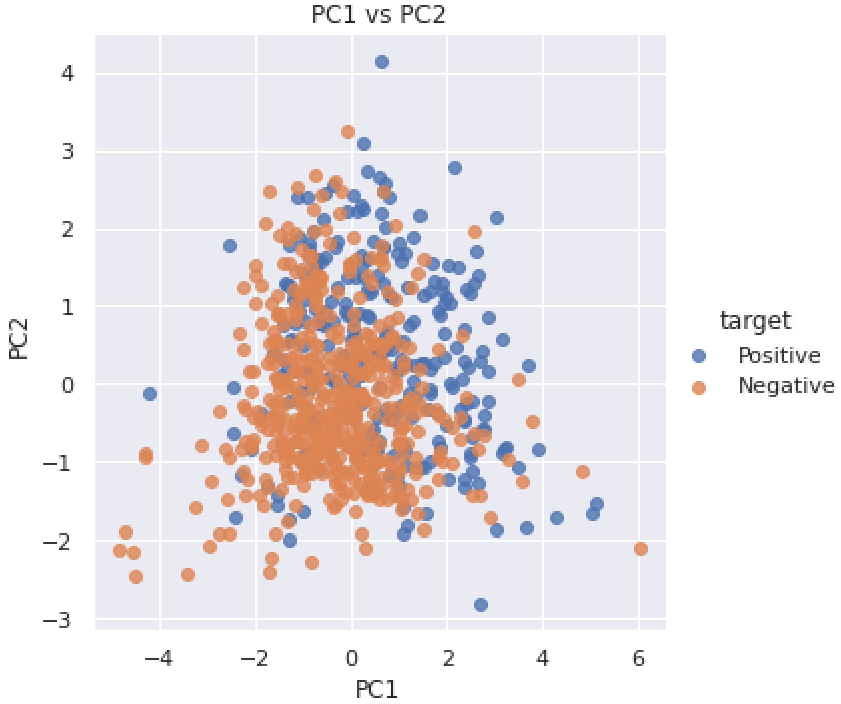


Question 3. Use a Python library to find the principal components and project the data on those components. Plot the projected data on the first and the second (principal components) PCs as a scatter plot. Details about using PCA can be found at: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

```
# get only the rows that correspond to the attributes
X = diabetes.values
# data preprocessing
X_scaled = StandardScaler().fit_transform(X)
# create PCA model
pca = PCA(n_components = 2)
# fit the model
features = pca.fit_transform(X_scaled)
# create dataframe with the new components
pca_df = pd.DataFrame(data = features, columns=['PC1', 'PC2'])

target_names = {0:'Negative', 1:'Positive'}
pca_df['target'] = y
pca_df['target'] = pca_df['target'].map(target_names)

# plotting the data on the first and second components
sns.set()
sns.lmplot(x='PC1', y='PC2', data=pca_df, hue='target', fit_reg=False, legend=True)
plt.title('PC1 vs PC2')
plt.show()
```



Task 3

Question 1. Perform pairwise comparison between the records in the dataset (publications.csv) to detect the duplicate records. To compare two records, follow the steps:

```
# loading the datasets
mapping = pd.read_csv(data_path + "pub_mappings.csv")
publications = pd.read_csv(data_path + "publications.csv")
pub_B1 = pd.read_csv(data_path + "publications_B1.csv")
pub_B2 = pd.read_csv(data_path + "publications_B2.csv")
```

```
publications
```

	id	title	authors	venue	year
0	304586	The WASA2 object-oriented workflow management ...	Gottfried Vossen, Mathias Weske	International Conference on Management of Data	1999
1	304587	A user-centered interface for querying distrib...	Isabel F. Cruz, Kimberly M. James	International Conference on Management of Data	1999
2	304589	World Wide Database-integrating the Web, CORBA...	Athman Bouguettaya, Boualem Benatallah, Lily H...	International Conference on Management of Data	1999
3	304590	XML-based information mediation with MIX	Chaitan Baru, Amarnath Gupta, Bertram Lud&#228...	International Conference on Management of Data	1999
4	304582	The CCUBE constraint object-oriented database ...	Alexander Brodsky, Victor E. Segal, Jia Chen, ...	International Conference on Management of Data	1999
...	...	...	...	...	...
995	conf/vldb/RamakrishnanR96	Modeling Design Versions	D. Janaki Ram, R. Ramakrishnan	VLDB	1996
996	conf/sigmod/BerchtoldK98	High-Dimensional Index Structures, Database Su...	Daniel A. Keim, Stefan Berchtold	SIGMOD Conference	1998

```
# exploring the dataset
all_combinations_df = pd.DataFrame()
# for every column
for col in publications.columns:
  list_left = []
  list_right = []
  # we make tuples with every combination
```

```
    for subset in itertools.combinations(publications[col], 2):
      list_left.append(subset[0])
      list_right.append(subset[1])
    all_combinations_df[col+"_1"] = list_left
    all_combinations_df[col+"_2"] = list_right

  all_combinations_df.head()
```

	id_1	id_2	title_1	title_2	authors_1	authors_2	venue_1	ve
0	304586	304587	The WASA2 object-oriented workflow management ...	A user-centered interface for querying distrib...	Gottfried Vossen, Mathias Weske	Isabel F. Cruz, Kimberly M. James	International Conference on Management of Data	Intern Conf Manag c
1	304586	304589	The WASA2 object-oriented workflow management ...	World Wide Database-integrating the Web, CORBA...	Gottfried Vossen, Mathias Weske	Athman Bouguettaya, Boualem Benatallah, Lily H...	International Conference on Management of Data	Intern Conf Manag c
2	304586	304590	The WASA2 object-oriented workflow management ...	XML-based information mediation with MIX	Gottfried Vossen, Mathias Weske	Chaitan Baru, Amarnath Gupta, Bertram Lud&#228...	International Conference on Management of Data	Intern Conf Manag c

a. Ignore the pub_id.

b. Use Levenshtein similarity for comparing the titles and computing the score (s_t)

```
# start time for the whole similarity
start = time.time()

lev = sm.levenshtein.Levenshtein()
#·compute·levenshtein·similarity·for·the·title·columns
list_distances = []
for index, row in all_combinations_df.iterrows():
  list_distances.append(lev.get_sim_score(string1=row["title_1"], string2=row["title_2"]))


all_combinations_df["s_t"] = list_distances
all_combinations_df
```

			...	with MIX
3	304586	304582	The WASA2 object-oriented workflow management ...	The CCUBE constraint object-oriented database ...
4	304586	304583	The WASA2 object-oriented workflow management ...	The Cornell Jaguar project: adding mobility to...
...	...	...	...	...
499495	conf/sigmod/BerchtoldK98	journals/sigmod/Winslett02a	High-Dimensional Index Structures, Database Su...	David DeWitt Speaks Out
499496	conf/sigmod/BerchtoldK98	conf/sigmod/AndersonAF98	High-Dimensional Index Structures, Database Su...	Oracle Rdb's Record Caching Model
499497	conf/sigmod/ChoALS03	journals/sigmod/Winslett02a	LockX: A System for Efficiently Querying Secur...	David DeWitt Speaks Out
499498	conf/sigmod/ChoALS03	conf/sigmod/AndersonAF98	LockX: A System for Efficiently Querying Secur...	Oracle Rdb's Record Caching Model
499499	journals/sigmod/Winslett02a	conf/sigmod/AndersonAF98	David DeWitt Speaks Out	Oracle Rdb's Record Caching Model

499500 rows × 11 columns

c. Use Jaro similarity to compare the values in the authors field and compute (s_a)

```
jaro = sm.jaro.Jaro()

list_distances_authors = []
# compute jaro similarity for the author columns
for index, row in all_combinations_df.iterrows():
  list_distances_authors.append(jaro.get_sim_score(string1=row["authors_1"], string2=row["


all_combinations_df["s_a"] = list_distances_authors
all_combinations_df
```

			...	with MIX
3	304586	304582	The WASA2 object-oriented workflow management ...	The CCUBE constraint object-oriented database ...
4	304586	304583	The WASA2 object-oriented workflow management ...	The Cornell Jaguar project: adding mobility to...
...	...	...	...	...
499495	conf/sigmod/BerchtoldK98	journals/sigmod/Winslett02a	High-Dimensional Index Structures, Database Su...	David DeWitt Speaks Out
499496	conf/sigmod/BerchtoldK98	conf/sigmod/AndersonAF98	High-Dimensional Index Structures, Database Su...	Oracle Rdb's Record Caching Model
499497	conf/sigmod/ChoALS03	journals/sigmod/Winslett02a	LockX: A System for Efficiently Querying Secur...	David DeWitt Speaks Out
499498	conf/sigmod/ChoALS03	conf/sigmod/AndersonAF98	LockX: A System for Efficiently Querying Secur...	Oracle Rdb's Record Caching Model
				Oracle Rdb's

d. Use the modified affine similarity for the conference (conf) attribute (s_c)

Model

```
def aff_sim (s1, s2, open_gap = 1, gap_ext = 0.1):
    aff_sim = sm.affine.Affine(gap_start = 1, gap_continuation = 0.1, \
                       sim_func = lambda s1, s2: (int(1 if s1 == s2 else 0)))
    return aff_sim.get_raw_score(s1, s2) / min(len(s1), len(s2))

list_distances_venues = []
# compute affine similarity for each row (venue columns)
```

```
for index, row in all_combinations_df.iterrows():
  list_distances_venues.append(aff_sim(s1=row["venue_1"], s2=row["venue_2"]))

all_combinations_df["s_c"] = list_distances_venues
all_combinations_df
```

			...	with MIX
3	304586	304582	The WASA2 object-oriented workflow management ...	The CCUBE constraint object-oriented database ...
4	304586	304583	The WASA2 object-oriented workflow management ...	The Cornell Jaguar project: adding mobility to...
...	...	...	...	...
499495	conf/sigmod/BerchtoldK98	journals/sigmod/Winslett02a	High-Dimensional Index Structures, Database Su...	David DeWitt Speaks Out

e. Use Match (1) / Mismatch (0) for the year (s_y)

```
all_combinations_df["s_y"] = np.where(all_combinations_df["year_1"] == all_combinations_df
all_combinations_df.head()
```

id_1	id_2	title_1	title_2	authors_1	authors_2	venue_1	ve
		The WASA2	A user-			International	Intern
		...	distrib...				

f. Use the formula rec_sim = 0.5 * s_t + 0.2 * s_a + 0.2 * s_c + 0.1 * s_y to combine the scores and compute the final score.

```
all_combinations_df["rec_sim"] = 0.5 * all_combinations_df["s_t"] + 0.2 * all_combinations_
all_combinations_df
```

	id_1	id_2	title_1	title_2
0	304586	304587	The WASA2 object-oriented workflow management ...	A user-centered interface for querying distrib...
1	304586	304589	The WASA2 object-oriented workflow management ...	World Wide Database-integrating the Web, CORBA...
2	304586	304590	The WASA2 object-oriented workflow management ...	XML-based information mediation with MIX
3	304586	304582	The WASA2 object-oriented workflow management ...	The CCUBE constraint object-oriented database ...
			The WASA2 ... management	The ... mobility

g. Report the records with rec_sim > 0.7 as duplicate records by storing the ids of both records in a list.

```
# records that have a rec_sim greater than 0.7
records_df = all_combinations_df[all_combinations_df['rec_sim'] > 0.7]
# get only the ids
ids_1 = records_df['id_1']
ids_2 = records_df['id_2']
# create a list of tuples containing both ids (records)
records = list(zip(ids_2, ids_1))
for record in records:
  print(record)
print(len(records))
```

```
('conf/sigmod/BouguettayaBH99', '304589')
('conf/sigmod/BaruGLMPVC99', '304590')
('journals/sigmod/JenningsNF98', '306112')
('conf/sigmod/BraumandlKK99', '304573')
('conf/sigmod/JarkeQBLMS99', '304568')
('conf/sigmod/RundensteinerCLCZNJLW99', '304579')
('304207', '304581')
('conf/sigmod/Suciu99', '304233')
('conf/sigmod/Mohan99', '304230')
('conf/sigmod/Keim99', '304219')
('375761', '304222')
```

```
('375668', '335383')
('journals/sigmod/PapianiWDN99', '333614')
('290607', '333616')
('381897', '333616')
('journals/sigmod/GruenwaldBDGSSV99', '333616')
('journals/sigmod/SouzaS99', '310063')
('journals/sigmod/ShethBS99', '310067')
('journals/tods/CasatiCPP99', '328996')
('journals/tods/Alagic99', '328943')
('journals/sigmod/Mackay99', '309852')
('journals/sigmod/SalgadoE03', '310062')
('277954', '277955')
('journals/tods/LiuHD98', '277628')
('conf/sigmod/ZaianeHLCC98', '276388')
('conf/sigmod/Toyama98', '276389')
('conf/sigmod/BaumannDFRW98', '276386')
('conf/sigmod/AmbiteABKMMMPT98', '276381')
('conf/sigmod/LiS98', '276374')
('conf/sigmod/Whelan98', '276368')
('conf/sigmod/LahiriJJC98', '276366')
('conf/sigmod/AndersonAF98', '276365')
('conf/sigmod/Chan98', '276362')
('conf/sigmod/Spiro98', '276359')
('conf/sigmod/WhiteCF98', '276355')
('conf/sigmod/BerchtoldK98', '276353')
('conf/sigmod/AndersonBKW98', '276347')
('conf/sigmod/LometW98', '276345')
('304193', '276340')
('conf/sigmod/FernandezFKLS98', '276341')
('conf/sigmod/ChaudhuriMN99', '304206')
('conf/sigmod/MankuRL99', '304204')
('335450', '304207')
('375692', '304210')
('conf/sigmod/IvesFFLW99', '304209')
('conf/sigmod/VitterW99', '304199')
('conf/sigmod/Wu99', '304202')
('conf/sigmod/LometT99', '304189')
('conf/sigmod/AdaliSS99', '304193')
('conf/sigmod/JagadishLMSV99', '304194')
('conf/sigmod/MamoulisP99', '304183')
('375727', '304184')
('conf/sigmod/AnkerstBKS99', '304187')
('390004', '290599')
('journals/sigmod/RossGN03', '290599')
('journals/sigmod/RossCGLLM01', '290599')
('journals/sigmod/RossHKRRSS01', '290599')
('journals/sigmod/RossFS02', '290599')
```

h. In the table pub_mappings.csv, you can find the actual mappings (the ids of the correct duplicate records). Compare the accuracy of this method by counting the number of duplicate records that you discovered correctly.

```
mapping
```

	idDBLP	idACM
0	conf/sigmod/SlivinskasJS01	375678
1	conf/sigmod/RinfretOO01	375669
2	conf/sigmod/DattaDRTV01	375780
3	conf/sigmod/BorkarDS01	375682
4	conf/sigmod/HanPDW01	375664
...	...	...
88	journals/tods/DyresonS98	288087
89	journals/tods/StolboushkinT98	288089
90	journals/tods/MamoulisP01	503101
91	journals/sigmod/KantM00	362091
92	journals/sigmod/OukselS99	309849

```
# transforming mappings to lists and creating
# a list of tuples too to compare with the records
mappings_1 = mapping["idDBLP"].tolist()
mappings_2 = mapping["idACM"].tolist()
mappings_tuples = list(zip(mappings_1, mappings_2))

# cast second column to string in order to compare both lists
mappings_transformed = []
for tupla in mappings_tuples:
  mappings_transformed.append((tupla[0], str(tupla[1])))

# computing the accuracy
accuracy = 0
for record in records:
  for map in mappings_transformed:
    if record[0] == map[0] and record[1] == map[1]:
      accuracy += 1

print("Accuracy: {}".format(accuracy/len(mapping)))
end = time.time()
```

```
Accuracy: 0.967741935483871
```

i. Record the running time of the method when processing the pairwise similarity between the 1000 records.

```
print("The running time is {} minutes".format((end-start)/60))
```

```
The running time is 7.309100862344106 minutes
```

Question 2. Repeat question1 but compare only the records from table publications_B1.csv with those in publications_B2.csv (do not compare the records that exist in the same file). That is, you will compare each record from the 500 records in the first table with all records in the second table. Compute the accuracy and the running time and compare the running time with

```
# exploring the dataset
pub_B1_B2 = pd.DataFrame()
for col1 in pub_B1:
  for col2 in pub_B2:
    if col1 == col2:
      list_products = list(product(pub_B1[col1], pub_B2[col2]))
      list_products_0 = [tupla[0] for tupla in list_products]
      list_products_1 = [tupla[1] for tupla in list_products]
      pub_B1_B2[col1+"_1"] = list_products_0
      pub_B1_B2[col2+"_2"] = list_products_1
pub_B1_B2
```

	id_1	id_2	title_1	title_2	authors_1	autl
0	304586	journals/sigmod/Mackay99	The WASA2 object-oriented workflow management ...	Semantic Integration of Environmental Models f...	Gottfried Vossen, Mathias Weske	D M
1	304586	conf/vldb/PoosalaI96	The WASA2 object-oriented workflow management ...	Estimation of Query-Result Distribution and it...	Gottfried Vossen, Mathias Weske	Vis P Ya Io

The WASA2

```
# start time for the whole similarity
start_2 = time.time()

lev_2 = sm.levenshtein.Levenshtein()

list_distances_2 = []
for index, row in pub_B1_B2.iterrows():
  list_distances_2.append(lev_2.get_sim_score(string1=row["title_1"], string2=row["title_2

pub_B1_B2["s_t"] = list_distances_2
```

The WASA2

```
jaro_2 = sm.jaro.Jaro()

list_distances_authors_2 = []
for index, row in pub_B1_B2.iterrows():
  list_distances_authors_2.append(jaro_2.get_sim_score(string1=row["authors_1"], string2=r

pub_B1_B2["s_a"] = list_distances_authors_2
```

Workshop on Krishna

```
list_distances_venues_2 = []
# compute affine similarity for each row (venue columns)
for index, row in pub_B1_B2.iterrows():
  list_distances_venues_2.append(aff_sim(s1=row["venue_1"], s2=row["venue_2"]))

pub_B1_B2["s_c"] = list_distances_venues_2
```

249996 362091 conf/sigmod/BerchtoldK98 performance Index Kant, Keim

```
pub_B1_B2["s_y"] = np.where(pub_B1_B2["year_1"] == pub_B1_B2["year_1"], 1, 0)
all_combinations_df.head()
```

	id_1	id_2	title_1	title_2	authors_1	authors_2	venue_1	ve
0	304586	304587	The WASA2 object-oriented workflow management ...	A user-centered interface for querying distrib...	Gottfried Vossen, Mathias Weske	Isabel F. Cruz, Kimberly M. James	International Conference on Management of Data	Intern Conf Manag c
1	304586	304589	The WASA2 object-oriented workflow management ...	World Wide Database-integrating the Web, CORBA...	Gottfried Vossen, Mathias Weske	Athman Bouguettaya, Boualem Benatallah, Lily H...	International Conference on Management of Data	Intern Conf Manag c
2	304586	304590	The WASA2 object-oriented workflow management ...	XML-based information mediation with MIX	Gottfried Vossen, Mathias Weske	Chaitan Baru, Amarnath Gupta, Bertram Lud&#228...	International Conference on Management of Data	Intern Conf Manag c
3	304586	304582	The WASA2 object-oriented workflow management ...	The CCUBE constraint object-oriented database ...	Gottfried Vossen, Mathias Weske	Alexander Brodsky, Victor E. Segal, Jia Chen, ...	International Conference on Management of Data	Intern Conf Manag c
				The				

```
pub_B1_B2["rec_sim"] = 0.5 * pub_B1_B2["s_t"] + 0.2 * pub_B1_B2["s_a"] + 0.2 * pub_B1_B2["
pub_B1_B2
```

	id_1	id_2	title_1	title_2	authors_1	
0	304586	journals/sigmod/Mackay99	The WASA2 object-oriented workflow management ...	Semantic Integration of Environmental Models f...	Gottfried Vossen, Mathias Weske	
1	304586	conf/vldb/PoosalaI96	The WASA2 object-oriented workflow management ...	Estimation of Query-Result Distribution and it...	Gottfried Vossen, Mathias Weske	
2	304586	conf/vldb/PalpanasSCP02	The WASA2 object-oriented workflow management ...	Incremental Maintenance for Non-Distributive A...	Gottfried Vossen, Mathias Weske	T Ric
3	304586	conf/vldb/GardarinGT96	The WASA2 object-oriented workflow management ...	Cost-based Selection of Path Expression Proces...	Gottfried Vossen, Mathias Weske	J
4	304586	conf/vldb/HoelS95	The WASA2 object-oriented workflow management ...	Benchmarking Spatial Join Operations with Spat...	Gottfried Vossen, Mathias Weske	E Ha
...	...	...	...	...	...	
249995	362091	conf/vldb/RamakrishnanR96	Workshop on performance and architecture of we...	Modeling Design Versions	Krishna Kant, Prasant Mohapatra	Ra
249996	362091	conf/sigmod/BerchtoldK98	Workshop on performance and architecture of we...	High-Dimensional Index Structures, Database Su...	Krishna Kant, Prasant Mohapatra	K
			Workshop	LockX: A		

```
from numpy.core.arrayprint import printoptions
# records that have a rec_sim greater than 0.7
records_df_2 = pub_B1_B2[pub_B1_B2['rec_sim'] > 0.7]
# get only the ids
ids_1_1 = records_df_2['id_1']
```