7/31/2018

Data Management, Warehousing Analytics CSCI 5408

Assignment 6: Analyzing data patterns and different classification methods

Sogra Bilal Memon
B00786252

Bhavya Chandrappa
B00781097

Table of Contents

1. Task Description

In the objective of this project was to implement different machine learning algorithms on a single data set so that the performance of each algorithm can be compared to that of the other. This project uses the language dataset from DSL 2014 workshop to train machine learning models to predict the language a sentence has been written in based on the occurrences or absence of certain words. The data set consists of two .txt files one for the testing data and the other for the training data.[1]

The process of achieving the predictions from these machine learning algorithms involves a number of steps. The data set has to downloaded and imported into the Anaconda Jupiter notebook. Followed by the process of feature extraction, feature selection and model training. These processes are encapsulated in a pipeline so that the training dataset can be rationalized and passed through the already trained model. The resulting prediction is then compared to the actual results to calculate the accuracy of the model.[2], [3]

The accuracy of a model varies based on multiple parameters such as the algorithm used and how well it is suited to the data as well as the features selected.[4] The variation of model accuracy has been shown based on the change in these parameters. A number of visualization techniques have been used to represent this data. In order to perform these predictions, visualizations and analysis a number of resources have been used. These resources include Python libraries like the sklearn, numpy, pandas and matplotlib.[5][6]–[8][9] The Anaconda Jupiter notebook provides a user friendly development environment for easy execution and debugging.[10]

2. Procedures Followed

2.1 Parsing dataset into a Dataframe

The dataset consists of sentences and the language this sentence belongs to. This data is not in a suitable format for training a classifier.

jupyter test2 Last Checkpoint: 14 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [228]:

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
```

In [229]:

```
df=pd.read_csv('train.txt' ,sep='\t', names=['Scentence','Language'] )
```

In [230]:

```
df
```

Out[230]:

	Scentence	Language
0	Зад думите "просто искам да се махна от Българ...	bg
1	Сега нещата там леко потръгнаха с усилията на ...	bg
2	Хърватският филм "Пътят на дините" на режисьор...	bg
3	Министърът на правосъдието на РС Джерард Селма...	bg
4	В крайна сметка войната между Севера и Юга дов...	bg
5	Новините на югозападна България Пушачите в пар...	bg
6	Преди два дни той и директорът на Историческия...	bg
7	Румъния ще бъде представена на игрите от колое...	bg
8	Ще си позволя да вмъкна един съвет, въдях че к...	bg
9	След съобщението на Папандреу за сформиране на...	bg
10	Съорганизатори на курса в Черна гора са филиал...	bg
11	Стопаните на котки често споделят, че когато и...	bg
12	Лошото е, че ти не ме позна, че аз съм твой съ...	bg
13	Единственото решение е кабелите да бъдат отдад...	bg
14	Новият бюджет трябва да спомогне за балансиран...	bg
15	Съюзът заяви, че не трябва да има политическа ...	bg
16	Висящи в нищото като усмивката на Чеширския ко...	bg
17	Сватбата на британската "Пепеляшка" е само сле...	bg
28	Във ваканционните месеци глътка въздух за инте...	bg
29	Това писмо бе изпратено до казионната БНТ 1. К...	bg
...	...	...
236105	"Champion Petfoods" специализируется на изгото...	xx
236106	Inicialment, assegura, havien de marxar diumen...	xx
236107	Hindi 'yung pupunta ka roon na wala kang alam ...	xx
236108	Pretekli teden je bil na političnem področju p...	xx
236109	Wala kasing pakialam ang presidential candidat...	xx
236110	Kinumpirma kahapon ng pamunuan ng Overseas Wor...	xx
236111	Правительство также согласилось с предложениям...	xx
236112	Okej, šef kuhinje priporoča, da jed, podobno k...	xx
236113	А когда Андрей Веревский сумеет наладить полно...	xx
236114	Ngunit hindi naman natagpuan ng mga sundalo an...	xx
236115	Сборная России по керлингу на колясках одержал...	xx
236116	Z novim letom jih k poročanju zavezuje zakon, ...	xx
236117	I possiblement sigui el dàtil un dels primers ...	xx
236118	По словам посла, он очень надеется на то, что ...	xx
236119	Andrej Bajuk se je rodil 18. oktobra 1943 v Lj...	xx
236120	Agad namang tumakas ang mga suspek habang dumu...	xx
236121	Для справедливости стоит отметить, что схожая ...	xx
236122	Nangako si Lorna na sasama ito sa mga susunod ...	xx
236123	Per consegüent, si no s'aconsegueix reunir un ...	xx
236124	Kaj je torej ta zločin, ki ga je storil vseobv...	xx
236125	Как сообщили представители службы гражданской ...	xx
236126	Za Slovenijo je bilo doslej najuspešnejše, saj...	xx
236127	Ayon kay Senador Gregorio Honasan, "prerogativ...	xx
236128	In kaj festival danes pomeni domačinom? "Tradi...	xx

Figure 1: The training dataset imported into a pandas Dataframe

2.2 Feature Extraction

The CountVectorizer() function has to be used to extract the features from the dataset.[11] Features are parameters that vary from one record to another effecting the resulting classification of the record. In this case the frequency of occurrence of each word in a given sentence are the features extracted and the language a sentence belongs to is the label used for classification.

```
In [398]: x_traincv=cv.fit_transform(df_x)

In [399]: x=x_traincv.toarray()

In [400]: from sklearn.feature_extraction.text import TfidfTransformer
          tf_transformer = TfidfTransformer(use_idf=False).fit(x)

In [401]: X_train_tf = tf_transformer.transform(x)

In [402]: X_train_tf.shape
Out[402]: (2600, 35996)

In [403]: a=X_train_tf.toarray()

In [404]: a
Out[404]: array([[0.14142136, 0.14142136, 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 ...,
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ],
                 [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                  0.        ]])
```

Figure 2: The training dataset imported into a pandas Dataframe

2.3 Sampling

This dataset is quite large consisting of 2000 records per language. Since this dataset has thirteen languages this means there are 26,0000 records in total.[1] In order to generate plots that show a clear separation of points, based on the language class the point belongs to, a smaller sample of points has been taken for plotting the graphs. The sample function from the pandas library has been used to select an equal number of points from each of the thirteen language classes.[5] These points from each of the thirteen languages have been chosen in random order. We wanted to select 200 points from each language this is achieved by setting the n parameter of the sample function to the total number of records to be returned and leaving all the other parameters at the default setting. When all parameters except the n value are given the default setting the function chooses an equal weightage of points for each of the classes.[5] In our case we need 200 records for each of the 13 values giving a total of 2600 returned records. Therefor the value of the n parameter must be set to 2600 to select 200 points from each language.

```
In [217]: df=df.sample(n=2600)

In [218]: df
Out[218]:
```

	Scentence	Language
37897	Mezi těmi, kdo by mohl před soudce předstoupit...	cz
191119	Mann pri prezentácii grafov totiž spojil reáln...	sk
129012	Министерот за регионален развој и јавни работи...	mk
203116	Kako to da posle dve godine postoji potreba za...	sr
111577	Ketua Persatuan Penyandang Cacat Indonesia (PP...	id
218820	On je rekao da je Rusija zainteresovana i za p...	sr
141240	'Mereka siapkan makalah dan juga menghias papa...	my
63884	El Ministerio de Salud de la Nación recomendó ...	es-AR
172358	Nesse sentido, o Papa vincou ontem a necessida...	pt-PT
22352	Ipak, sjenku na cjelokupnu ceremoniju bacila j...	bs
74382	La presidenta de la Comunidad de Madrid, la po...	es-ES
151297	PENGALAMAN pahit terjerumus ke saingan Liga Pe...	my
65130	El estilo, copiado por gran parte de las admir...	es-AR
129551	Во овие два ипол месеци ни одзеде многу енерги...	mk
216146	Bivši jugoslovenski predsednik Slobodan Miloše...	sr
163176	O aumento previsto se aplica aos vencimentos d...	pt-BR
29108	U trenutku Bulatovicevog ubistva na neki nacin...	bs
191703	Viac však ide podľa nej v tejto podmienke o to...	sk
1177	Горнооряховският кмет не иска боклуците на Пол...	bg
100406	Na primjeru Jadrantekstila i Vidoševićeve majk...	hr
84962	Sin embargo, ha insistido en que a partir de e...	es-ES
136495	Според директорката на Филхармонијата Маја Чан...	mk
...	...	...
68224	La norma puntualiza que a los fines de determi...	es-AR
101365	Ta će tvrtka svoje dućane pod nazivom XYZ imat...	hr
229037	Sa ulat ng North Cotabato Provincial Disaster ...	xx
214829	Njihov san je da se Slovenija plasira u drugu ...	sr
64143	Se mostró como garante del respeto de Républic...	es-AR
139720	Bergantung kepada pakej pilihan pelanggan, ses...	my
115513	Disamping itu, tiga pemain Persmin yang ikut p...	id
155370	Os economistas também chamam atenção para a po...	pt-BR
85130	El Puerto rechazó ayer contestar a Ecologistas...	es-ES
176479	Na entrevista ao diário desportivo, a 13 de no...	pt-PT
172113	Para cumprirem os seus objectivos não precisar...	pt-PT
184308	" "O problema é falta de acesso com facilidade...	pt-PT
158997	Aos 45 minutos, o lento Everton perdeu uma boa...	pt-BR
58844	La Serie A italiana con 1532 millones de euros...	es-AR
25999	Znali smo, naravno, da bi bilo kakav otpor Zek...	bs
147620	'' Ketiadaan Samaranch memberikan peluang kepa...	my
134551	Алексеј Втори ја поддржуваше МПЦ Во Македонија...	mk
102257	Zemlje istočne Europe i bivšeg Sovjetskog Save...	hr

2600 rows × 2 columns

Figure 3: running query after extraction

2.4 Feature Selection

We have used chi square to select the features since the desired number of features can be selected with best scores using the chi square statistics. The training dataset is divided to data and target. Using SelectKBest function, we select best features. The best features are displayed using fit_transform function.

In [421]:
```
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.datasets import load_iris
iris = load_iris()
x_train, y_train = iris.data, iris.target# Select 3 top features
selection = SelectKBest(chi2, k=3)
```

In [422]:
```
selection.fit_transform(x_train, y_train)
```

Out[422]:
```
array([[5.1, 1.4, 0.2],
       [4.9, 1.4, 0.2],
       [4.7, 1.3, 0.2],
       [4.6, 1.5, 0.2],
       [5. , 1.4, 0.2],
       [5.4, 1.7, 0.4],
       [4.6, 1.4, 0.3],
       [5. , 1.5, 0.2],
       [4.4, 1.4, 0.2],
       [4.9, 1.5, 0.1],
       [5.4, 1.5, 0.2],
       [4.8, 1.6, 0.2],
       [4.8, 1.4, 0.1],
       [4.3, 1.1, 0.1],
       [5.8, 1.2, 0.2],
       [5.7, 1.5, 0.4],
       [5.4, 1.3, 0.4],
       [5.1, 1.4, 0.3],
       [5.7, 1.7, 0.3],
```

Figure 4: The three best features

```
In [299]: y_train=df["Language"]
          y_train
          #ay= y_train.toarray

Out[299]: 92107         hr
          215429        sr
          196971        sk
          118461        id
          57203      es-AR
          168426     pt-PT
          28027         bs
          212969        sr
          147004        my
          91292         hr
          154313     pt-BR
          156602     pt-BR
          128438        mk
          217649        sr
          138243        my
          53555      es-AR
          65562      es-AR
          26386         bs
          67565      es-AR
          109688        id
          216743        sr
          118508        id
          16463         bg
          186554        sk
          144376        my
          146992        my
          217614        sr
          138633        my
          79182      es-ES
          119142        id
                      ...
          35282         cz
```

Figure 5. Language column is taken as training dataset

The whole dataset is divided into training and testing data. Training data is fitted into df_x and df_y. The testing data is fitted into dft. The number of columns, 2600 is randomly selected from the dataset as training and testing. The training dataset is utilized to train the pipeline model and testing data is used to predict accuracy and confusion matrix of the model. Detailed process is described below. The test data is splitted into two dataframes, one dataframe consisting Sentence column, other column consisting Language dataset.

```
In [124]: dft=pd.read_csv('C:/Users/bhavy/Downloads/DSL-Task-master/DSL-Task-master/data/DSLCC-v2.0/test/test.txt' ,sep='\t', names=['Scente
```

```
In [125]: dft=df.sample(n=2600)
```

```
In [126]: dft
```

Out[126]:

	Scentence	Language
99074	Puniti će se mađarska i norveška blagajna na š...	hr
155974	Já a Copa do Brasil, prioridade tricolor no pr...	pt-BR
128712	(Од лево) палестинскиот лекар со бугарско држа...	mk
182574	Porém, esta série inexplicável de erros, sempr...	pt-PT
59875	A sólo unas semanas del Salón de Ginebra 2011,...	es-AR
123999	Милошевиќ умре на 11 март, ден пред денот на к...	mk
170636	Francisco Rodrigues Santos, da Associação de E...	pt-PT
66511	Y en sintonía con los instrumentos contracícli...	es-AR
10735	По случая с лекарите в Горна Оряховица Цветано...	bg
191349	Keď budú existovať dve strany, v istom zmysle ...	sk
11348	Румъния е на последно място в ЕС по размер на ...	bg
173279	Esta ronda de audições decorre depois do Presi...	pt-PT
58923	Racanelli pidió la libertad de ambos, pero le ...	es-AR
165737	Salvador, 01 de março de 2011 Repercussão de s...	pt-BR

Figure 6: Test data

2600 rows × 2 columns

```
In [127]: dft_x=dft["Scentence"]
          dft_y = dft["Language"]
```

```
In [128]: dft_x
```

```
Out[128]: 99074     Puniti će se mađarska i norveška blagajna na š...
          155974    Já a Copa do Brasil, prioridade tricolor no pr...
          128712    (Од лево) палестинскиот лекар со бугарско држа...
          182574    Porém, esta série inexplicável de erros, sempr...
          59875     A sólo unas semanas del Salón de Ginebra 2011,...
          123999    Милошевиќ умре на 11 март, ден пред денот на к...
          170636    Francisco Rodrigues Santos, da Associação de E...
          66511     Y en sintonía con los instrumentos contracícli...
          10735     По случая с лекарите в Горна Оряховица Цветано...
          191349    Keď budú existovať dve strany, v istom zmysle ...
          11348     Румъния е на последно място в ЕС по размер на ...
          173279    Esta ronda de audições decorre depois do Presi...
          58923     Racanelli pidió la libertad de ambos, pero le ...
          165737    Salvador, 01 de março de 2011 Repercussão de s...
          149757    Beliau berkata lebih mudah untuk mengawal golo...
          112616    Puncaknya, JC bertugas menggelar kongres pada ...
          92963     Dok smo punili krmeno skladište vodom, naš str...
          196758    Banka však nebije na poplach a argumentuje, že...
          228974    Iz suhoparne formulacije referendumskega vpraš...
          45634     Je neuvěřitelné jak si všichni lidi přejí aby ...
          105098    Terlepas dari siapa yang menyebarkan, yang jel...
          137007    Но, само еден ден по потпишувањето на меморанд...
          164622    Unidade Modular do Beira Rio retoma atendiment...
          114109    Puluhan tenaga bakti Pemadam Kebakaran Krueng ...
          11564     Ургвайците обаче не само участват, но и отива...
          32776     Programi za pripremnost u slučaju vanrednih st...
          10151     Според съобщение на Ройтерс, Лимай е заявил по...
          188490    Roky trvajúce rokovania na expertnej úrovni o ...
```

Figure 7: Displaying one set of Dataframe

```
In [129]: dft_y

Out[129]: 99074         hr
          155974     pt-BR
          128712        mk
          182574     pt-PT
          59875      es-AR
          123999        mk
          170636     pt-PT
          66511      es-AR
          10735         bg
          191349        sk
          11348         bg
          173279     pt-PT
          58923      es-AR
          165737     pt-BR
          149757        my
          112616        id
          92963         hr
          196758        sk
          228974        xx
          45634         cz
          105098        id
          137007        mk
          164622     pt-BR
          114109        id
          11564         bg
          32776         bs
          10151         bg
          188490        sk
          98662         hr
          139203        my
                     ...
          96013         hr
          125873        mk
```

Figure 8: Displaying Language dataframe

2.5 Pipelining

The pipeline is consisted of vectorizer, classifier and feature selection. The features are counted by vectorizer and selected and reduced by Dimensionality reduction algorithm and the classifier is applied on the selected features.

```
In [130]: from sklearn.pipeline import FeatureUnion
          from sklearn.pipeline import Pipeline
          from sklearn.decomposition import TruncatedSVD
          from sklearn.linear_model import LogisticRegression
          ##pipeline = Pipeline([('clf', LinearSVC(random_state=0))])
          ##pipeline.fit(x_train)

          lrp = Pipeline([
              ('vect', CountVectorizer()),
              ('svd', TruncatedSVD(n_components=10)),
              ##('tfid',TfidfTransformer()),
                                  ('lr',LogisticRegression())
                                  ])
```

```
In [131]: lrp.steps

Out[131]: [('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                    dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                    lowercase=True, max_df=1.0, max_features=None, min_df=1,
                    ngram_range=(1, 1), preprocessor=None, stop_words=None,
                    strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                    tokenizer=None, vocabulary=None)),
           ('svd', TruncatedSVD(algorithm='randomized', n_components=10, n_iter=5,
                   random_state=None, tol=0.0)),
           ('lr',
            LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                      intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                      penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                      verbose=0, warm_start=False))]
```

Figure 9: Example of a Pipeline model

2.6 Training, Testing and Evaluating accuracy of the model

The pipeline model is then trained on the training dataset df_x and df_y and is predicted on the testing data that is different to training data.

```
In [132]: model = lrp.fit(df_x,df_y)

In [133]: model.predict(dft_x)

Out[133]: array(['cz', 'pt-BR', 'mk', ..., 'sk', 'pt-PT', 'es-ES'], dtype=object)
```

Figure 10: Pipeline model result

3. Classification Algorithm

3.1 Scatter plot of training data

Output:

Code:

```
In [414]: import matplotlib.pyplot as plt
```

```
In [420]: N = 50
          x = np.random.rand(N)
          y = np.random.rand(N)
          colors = np.random.rand(N)
          area = (30 * np.random.rand(N))**2  # 0 to 15 point radii

          plt.scatter(x_train[:,0], y_train, c=x_train[:,1], s=area, alpha=0.5)
          plt.show()
```

Figure 11: Code to scatter plot training data

The above shows the Scatter plot code on how x_train[:,0] is plotted against y_train where the x_train[:,1] are depicted as colors. The plt.show() displays the graph that is shown below. From the graph it is seen that second column of the training data is displayed in color and the first column of the data is depicted in x-axis.

Visualization:

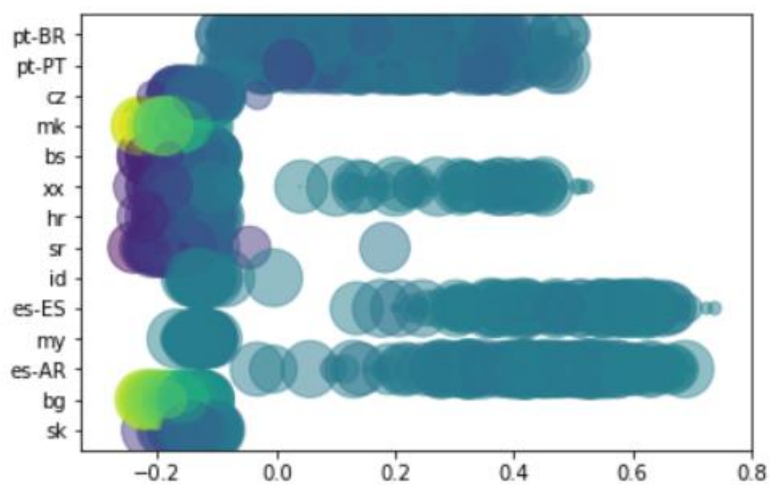

Figure 12: Scatter plot visualization

3.2 Linear SVM

Pipeline Code:

Linear Supervised Model classifier is used to classify the features. Each feature is varied with the value. The pipeline is consisted of CountVectorizer, TruncatedSVD and

LinearSVC. The model is fitted with training data and predicted with testing data. The output is displayed in one dimensional array.

```
In [137]: from sklearn.pipeline import FeatureUnion
          from sklearn.pipeline import Pipeline
          from sklearn.decomposition import TruncatedSVD
          from sklearn.svm import LinearSVC
          ##pipeline = Pipeline([('clf', LinearSVC(random_state=0))])
          ##pipeline.fit(x_train)

          lsvc = Pipeline([
              ('vect', CountVectorizer()),
              ('svd', TruncatedSVD(n_components=10)),
              ##('tfid',TfidfTransformer()),
                                ('lr',LinearSVC())
                                ])

In [138]: lsvc.steps

Out[138]: [('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                   dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                   lowercase=True, max_df=1.0, max_features=None, min_df=1,
                   ngram_range=(1, 1), preprocessor=None, stop_words=None,
                   strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                   tokenizer=None, vocabulary=None)),
           ('svd', TruncatedSVD(algorithm='randomized', n_components=10, n_iter=5,
                  random_state=None, tol=0.0)),
           ('lr', LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
                intercept_scaling=1, loss='squared_hinge', max_iter=1000,
                multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
                verbose=0))]

In [139]: model1 = lsvc.fit(df_x,df_y)

In [140]: model1.predict(dft_x)

Out[140]: array(['cz', 'pt-BR', 'mk', ..., 'xx', 'pt-BR', 'es-ES'], dtype=object)
```

Figure 13: Pipeline model prediction for Linear SVM

Accuracy:

The accuracy score when Linear SVC is implemented in pipeline model results in 0.57

```
In [141]: accuracy_score(dft_y,lsvc.predict(dft_x), normalize=True, sample_weight=None) #svc

Out[141]: 0.5746153846153846
```

Figure 14: Pipeline model accuracy for Linear SVM

```
In [142]: cm1=confusion_matrix(dft_y,lsvc.predict(dft_x))
          fg = plt.figure()
          axi = fg.add_subplot(111)
          cx = axi.matshow(cm1)
          plt.title('Linear SVC')
          fg.colorbar(cx)
          #ax.set_xticklabels([''] + labels)
          #ax.set_yticklabels([''] + labels)
          #plt.xlabel('Predicted')
          #plt.ylabel('True')
          plt.show()
```

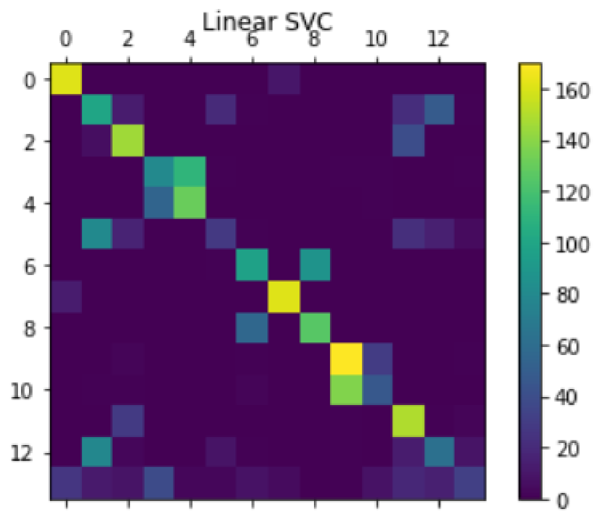

Figure 15: Pipeline model confusion matrix for Linear SVM

3.3 Logistic Regression

Pipeline Code:

Logistic Regression classifier is used to classify the features. Each feature is plotted with ita value to form a line. The pipeline is consisted of CountVectorizer, TruncatedSVD and Logistic Regression. The model is fitted with training data and predicted with testing data. The output is displayed in one dimensional array.

```
In [130]: from sklearn.pipeline import FeatureUnion
          from sklearn.pipeline import Pipeline
          from sklearn.decomposition import TruncatedSVD
          from sklearn.linear_model import LogisticRegression
          ##pipeline = Pipeline([('clf', LinearSVC(random_state=0))])
          ##pipeline.fit(x_train)

          lrp = Pipeline([
              ('vect', CountVectorizer()),
              ('svd', TruncatedSVD(n_components=10)),
              ##('tfid',TfidfTransformer()),
                                ('lr',LogisticRegression())
                                ])
```

```
In [131]: lrp.steps
```

```
Out[131]: [('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                    dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                    lowercase=True, max_df=1.0, max_features=None, min_df=1,
                    ngram_range=(1, 1), preprocessor=None, stop_words=None,
                    strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                    tokenizer=None, vocabulary=None)),
           ('svd', TruncatedSVD(algorithm='randomized', n_components=10, n_iter=5,
                   random_state=None, tol=0.0)),
           ('lr',
            LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                     intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                     penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                     verbose=0, warm_start=False))]
```

Figure 16: The training dataset imported into a pandas Dataframe

```
In [132]: model = lrp.fit(df_x,df_y)
```

```
In [133]: model.predict(dft_x)
```

```
Out[133]: array(['cz', 'pt-BR', 'mk', ..., 'sk', 'pt-PT', 'es-ES'], dtype=object)
```

Figure 17: Logistic Regression pipeline model prediction

Accuracy:

The accuracy of the Logistic Regression pipeline model is 0.56

```
In [134]: from sklearn.metrics import accuracy_score
```

```
In [135]: accuracy_score(dft_y,lrp.predict(dft_x), normalize=True, sample_weight=None) #logistic
```

```
Out[135]: 0.5676923076923077
```

Figure 18: Accuracy for Logistic Regression pipeline model

Output: Confusion matrix

```
In [136]: from sklearn.metrics import confusion_matrix
          cm=confusion_matrix(dft_y,lrp.predict(dft_x)) #logistic
          fg = plt.figure()
          axi = fg.add_subplot(111)
          cx = axi.matshow(cm)
          plt.title('Logistic Regression')
          fg.colorbar(cx)
          #ax.set_xticklabels([''] + labels)
          #ax.set_yticklabels([''] + labels)
          #plt.xlabel('Predicted')
          #plt.ylabel('True')
          plt.show()
```

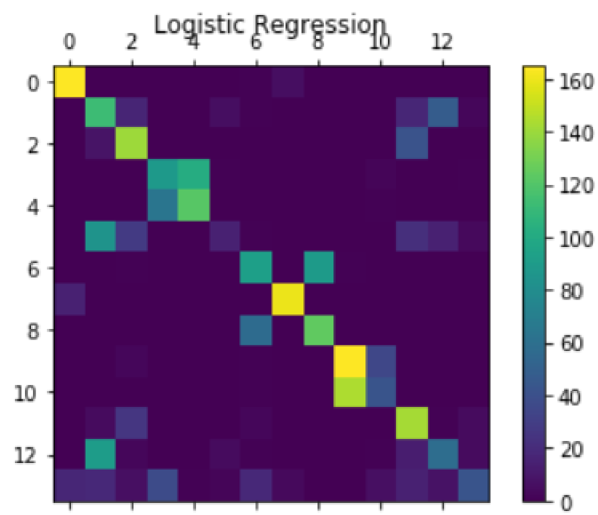

Figure 19: Confusion matrix for Logistic Regression pipeline model

3.4 Decision Tree

Pipeline Code:

Here, the decision Tree classifier is used to classify the features. The pipeline is consisted of CountVectorizer, TruncatedSVD and Decision Tree. The model is fitted with training data and predicted with testing data. The output is displayed in one dimensional array.

```
In [155]: from sklearn.tree import DecisionTreeClassifier
          dtc= Pipeline([
              ('vect', CountVectorizer()),
              ('svd', TruncatedSVD(n_components=10)),
              ##('tfid',TfidfTransformer()),
                                  ('dtc', DecisionTreeClassifier())
                                  ])
```

```
In [156]: dtc.steps

Out[156]: [('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                   dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                   lowercase=True, max_df=1.0, max_features=None, min_df=1,
                   ngram_range=(1, 1), preprocessor=None, stop_words=None,
                   strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                   tokenizer=None, vocabulary=None)),
           ('svd', TruncatedSVD(algorithm='randomized', n_components=10, n_iter=5,
                  random_state=None, tol=0.0)),
           ('dtc',
            DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                        splitter='best'))]
```

Figure 20: Decision Tree classifier pipeline model

```
In [157]: model4 = dtc.fit(df_x,df_y)
```

```
In [158]: model4.predict(dft_x)

Out[158]: array(['hr', 'pt-BR', 'mk', ..., 'hr', 'pt-PT', 'es-ES'], dtype=object)
```

Figure 21: Decision Tree classifier pipeline model prediction

Accuracy:

The accuracy of Decision Tree classifier is 1.0

```
In [159]: accuracy_score(dft_y,dtc.predict(dft_x), normalize=True, sample_weight=None)

Out[159]: 1.0
```

Figure 22: Decision Tree classifier pipeline model accuracy

Confusion Matrix:

```
In [160]: cm4=confusion_matrix(dft_y,dtc.predict(dft_x))
fg = plt.figure()
axi = fg.add_subplot(111)
cx = axi.matshow(cm4)
plt.title('Decision Tree Classifier')
fg.colorbar(cx)
#ax.set_xticklabels([''] + labels)
#ax.set_yticklabels([''] + labels)
#plt.xlabel('Predicted')
#plt.ylabel('True')
plt.show()
```

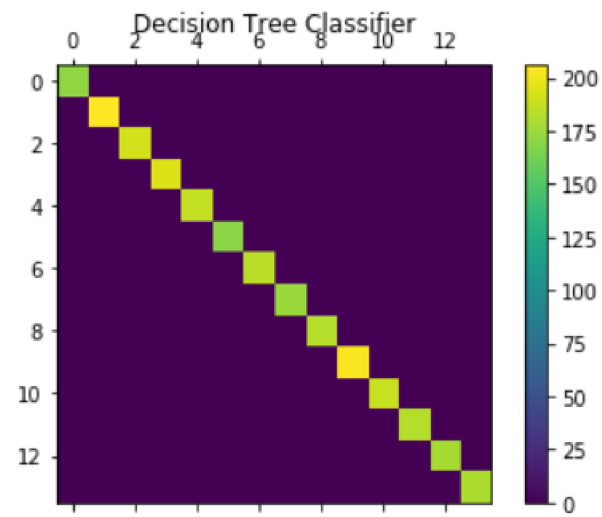

Figure 23: Decision Tree classifier pipeline model confusion matrix

3.5 Naïve Bayes

Pipeline Code:
Two pipelines named mnb and bnb are created where in mnb consists of CountVectorizer, SelectKBest and MultinomialNB classifier as shown in the below code. Bnb is created based on CountVectorizer, TruncatedSVD and Burnoulli NB. The model is predicted with the test data for both mnb and bnb.

```
In [143]: from sklearn.pipeline import FeatureUnion
          from sklearn.pipeline import Pipeline
          from sklearn.decomposition import TruncatedSVD
          from sklearn.naive_bayes import MultinomialNB
          ##pipeline = Pipeline([('clf', LinearSVC(random_state=0))])
          ##pipeline.fit(x_train)

          mnb = Pipeline([
              ('vect', CountVectorizer()),
              ('svd', SelectKBest()),
              ##('tfid',TfidfTransformer()),
                                  ('mnb',MultinomialNB())
                                  ])

In [144]: mnb.steps

Out[144]: [('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                    dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                    lowercase=True, max_df=1.0, max_features=None, min_df=1,
                    ngram_range=(1, 1), preprocessor=None, stop_words=None,
                    strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                    tokenizer=None, vocabulary=None)),
           ('svd',
            SelectKBest(k=10, score_func=<function f_classif at 0x000001E1082B81E0>)),
           ('mnb', MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True))]
```

Figure 24: Multinomial Naïve Bayes pipeline model

```
In [145]: model2 = mnb.fit(df_x,df_y)

In [146]: model2.predict(dft_x)

Out[146]: array(['bs', 'pt-BR', 'bs', ..., 'bs', 'pt-BR', 'es-ES'], dtype='<U5')
```

Figure 25: Multinomial Naïve Bayes pipeline model prediction

Accuracy:

The accuracy for the Multinomial Naïve Bayes is 0.26

```
In [147]: accuracy_score(dft_y,mnb.predict(dft_x), normalize=True, sample_weight=None) #mnb

Out[147]: 0.2684615384615385
```

Figure 26: Multinomial Naïve Bayes pipeline model accuracy

```
In [148]: cm2=confusion_matrix(dft_y,mnb.predict(dft_x))
          fg = plt.figure()
          axi = fg.add_subplot(111)
          cx = axi.matshow(cm2)
          plt.title('Mutlinomial MNB')
          fg.colorbar(cx)
          #ax.set_xticklabels([''] + labels)
          #ax.set_yticklabels([''] + labels)
          #plt.xlabel('Predicted')
          #plt.ylabel('True')
          plt.show()
```

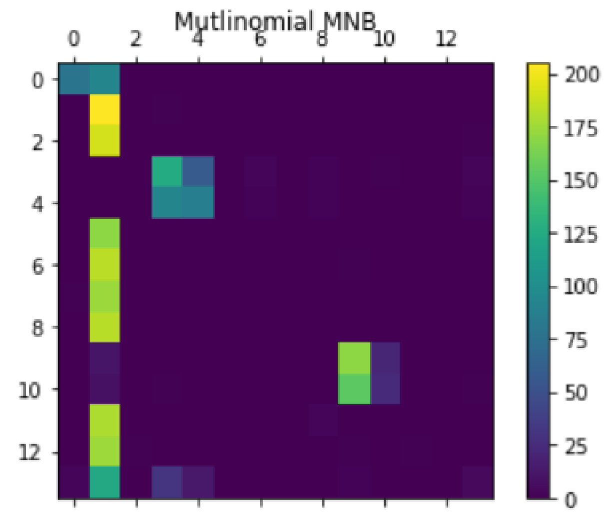

Figure 27: Multinomial Naïve Bayes pipeline model confusion matrix

3.6 Naïve Bayes – BNB

Pipeline:

Below shows the code for Bernoulli Naïve Bayes classifier that is fitted into pipeline.

```
In [149]: from sklearn.pipeline import FeatureUnion
          from sklearn.pipeline import Pipeline
          from sklearn.decomposition import TruncatedSVD
          from sklearn.naive_bayes import BernoulliNB
          ##pipeline = Pipeline([('clf', LinearSVC(random_state=0))])
          ##pipeline.fit(x_train)

          bnb= Pipeline([
              ('vect', CountVectorizer()),
              ('svd', TruncatedSVD(n_components=10)),
              ##('tfid',TfidfTransformer()),
                                ('mnb',BernoulliNB())
                                ]) #bnb
```

```
In [150]: bnb.steps

Out[150]: [('vect', CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                    dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                    lowercase=True, max_df=1.0, max_features=None, min_df=1,
                    ngram_range=(1, 1), preprocessor=None, stop_words=None,
                    strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                    tokenizer=None, vocabulary=None)),
           ('svd', TruncatedSVD(algorithm='randomized', n_components=10, n_iter=5,
                   random_state=None, tol=0.0)),
           ('mnb',
            BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True))]
```

Figure 28: Bernoulli Naïve Bayes pipeline model

```
In [151]: model3 = bnb.fit(df_x,df_y)
```

```
In [152]: model3.predict(dft_x)

Out[152]: array(['cz', 'pt-BR', 'bg', ..., 'sk', 'pt-BR', 'es-ES'], dtype='<U5')
```

Figure 29: Bernoulli Naïve Bayes pipeline model prediction

```
In [153]: accuracy_score(dft_y,bnb.predict(dft_x), normalize=True, sample_weight=None) #bnb

Out[153]: 0.4765384615384615
```

Figure 30: Bernoulli Naïve Bayes pipeline model accuracy

```
In [154]: cm3=confusion_matrix(dft_y,bnb.predict(dft_x))
          fg = plt.figure()
          axi = fg.add_subplot(111)
          cx = axi.matshow(cm3)
          plt.title('Burnoulli NB')
          fg.colorbar(cx)
          #ax.set_xticklabels(['']  + labels)
          #ax.set_yticklabels(['']  + labels)
          #plt.xlabel('Predicted')
          #plt.ylabel('True')
          plt.show()
```

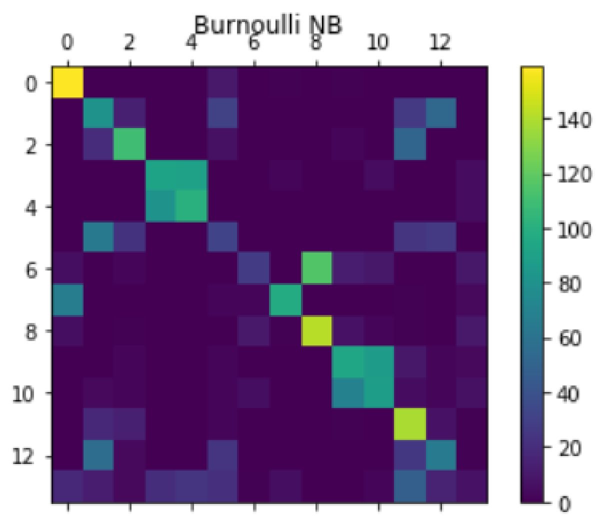

Figure 31: Bernoulli Naïve Bayes pipeline model confusion matrix

3.7 Comparison of accuracy

Classifier	Accuracy
Linear SVC	0.57
Logistic Regression	0.56
Decision Tree Classifier	1.0
Multinomial Naïve Bayes	0.26
Bernoulli Naïve Bayes	0.47

Table 1: Accuracy Comparision table

Based on the classifiers' accuracy, we can say that Decision Tree classifier gives the best accurate results compared to other classifiers. The Linear SVC and Logistic Regression give almost close accurate results after Decision Tree Classifier.

3.8 Impact of feature selection

The features that are selected and extracted from the given text contains noisy data that makes harder to plot the values. The noisy data can be viewed in the array when the Sentence text is mapped to an array.

4. Code Submission

The code has been submitted on both brightspace and on Github. The Github link has been given below.
https://github.com/SograMemon/DataWarehouse-A2.git

5. References

[1] "Dataset from DSL 2014 workshop." 2015.

[2] "CSCI5408 A6 tutorial - CSCI5408 - Data Mgmt, Warhsng Analytics (Sec 1) - 2018 Summer." [Online]. Available: https://dal.brightspace.com/d2l/le/content/73925/viewContent/1017316/View. [Accessed: 01-Aug-2018].

[3] "CSCI 5408 - Assignment 6 - CSCI5408 - Data Mgmt, Warhsng Analytics (Sec 1) - 2018 Summer." [Online]. Available: https://dal.brightspace.com/d2l/le/content/73925/viewContent/1015668/View. [Accessed: 01-Aug-2018].

[4] "Lecture 6 - ML - CSCI5408 - Data Mgmt, Warhsng Analytics (Sec 1) - 2018 Summer." [Online]. Available: https://dal.brightspace.com/d2l/le/content/73925/viewContent/1003939/View. [Accessed: 01-Aug-2018].

[5] "pandas.DataFrame.sample — pandas 0.23.3 documentation." [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sample.html. [Accessed: 01-Aug-2018].

[6] "NumPy Reference — NumPy v1.15 Manual." [Online]. Available: https://docs.scipy.org/doc/numpy/reference/. [Accessed: 01-Aug-2018].

[7] "pandas.read_csv — pandas 0.23.3 documentation." [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html. [Accessed: 01-Aug-2018].

[8] "scikit-learn: machine learning in Python — scikit-learn 0.19.2 documentation." [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 01-Aug-2018].

[9] "Matplotlib: Python plotting — Matplotlib 2.2.2 documentation." [Online]. Available: https://matplotlib.org/. [Accessed: 01-Aug-2018].

[10] "Downloads - Anaconda." [Online]. Available: https://www.anaconda.com/download/#macos. [Accessed: 01-Aug-2018].

[11] "sklearn.feature_extraction.text.CountVectorizer — scikit-learn 0.19.2 documentation." [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Accessed: 01-Aug-2018].

[12] "sklearn.metrics.accuracy_score" *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html. [Accessed: 02-Aug-2018].

[13] “Feature Union with Heterogeneous Data Sources” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/auto_examples/hetero_feature_union.html#sphx-glr-auto-examples-hetero-feature-union-py. [Accessed: 02-Aug-2018].

[14] “sklearn.naive_bayes.BernoulliNB” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB. [Accessed: 02-Aug-2018].

[15] “sklearn.naive_bayes.MultinomialNB” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB. [Accessed: 02-Aug-2018].

[16] “sklearn.tree.DecisionTreeClassifier” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier. [Accessed: 02-Aug-2018].

[17] “sklearn.svm.LinearSVC” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC. [Accessed: 02-Aug-2018].

[18] “1.1. Generalized Linear Models” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression. [Accessed: 02-Aug-2018].

[19] “sklearn.decomposition.PCA” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA. [Accessed: 02-Aug-2018].

[20] “Matplotlib scatterplot; colour as a function of a third variable,” *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/8202605/matplotlib-scatterplot-colour-as-a-function-of-a-third-variable. [Accessed: 02-Aug-2018].

[21] “Using sklearn Linear Regression and PCA in a single Pipeline,” *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/48549635/using-sklearn-linear-regression-and-pca-in-a-single-pipeline. [Accessed: 02-Aug-2018].

[22] “Working With Text Data” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html. [Accessed: 02-Aug-2018].

[23] “sklearn.metrics.confusion_matrix” *1.4. Support Vector Machines - scikit-learn 0.19.1 documentation*. [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html. [Accessed: 02-Aug-2018].