Lecture#11

Recitation Exercises

Text Classification on Scikit-learn

- Classify the Wikipedia 300 dataset (150 articles about Video games, 150 about Programming) using machine learning.

- For text classification the bag-of-words approach where you convert an article to word counts are typically used. An improvement is TF-IDF (Term Frequency-Inverse Document Frequency) which is used to convert from word counts to word frequencies.

Text Classification on Scikit-learn

- TF-IDF is especially useful if the size of the articles varies a lot. Suitable algorithms for text classification are Multinomial Naïve Bayes (MultinomialNB) and Support Vector Machines with linear kernels (LinearSVC).

https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Text Classification on Scikit-learn

- a) Classify the dataset using MultinomailNB and LinearSVC with the bag-of-words approach.
- b) Evaluate classification accuracy on the same data as used for training the algorithms.
- c) Evaluate classification accuracy using 10-fold cross validation.
- d) Use TF-IDF to convert from word counts to word frequencies.
- e) Does TF-IDF improve classification accuracy when using cross-validation?

Text Classification on Scikit-learn

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
import csv
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn import metrics
from sklearn import svm
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

In [2]:
```
wikipedia=pd.read_csv("C:\\Users \\Desktop\\wikipedia_300\\wikipedia_300\\wikipedia_300.csv")
wikipedia.head()
```

Text Classification on Scikit-learn

In [3]:
```
modelSVC = LinearSVC()
modelNB  = MultinomialNB()

wikipedia.shape
```
Out[3]: (300, 2)

In [4]:
```
wikipedia_y=wikipedia["Category"]
wikipedia_x=wikipedia["Text"]
```

Text Classification on Scikit-learn

In [4]:
```
wikipedia_y=wikipedia["Category"]
wikipedia_x=wikipedia["Text"]
```

In [5]:
```
cv=CountVectorizer()
bag_of_words =cv.fit_transform(wikipedia_x)
```

In [16]:
```
x_train,x_test,y_train,y_test =train_test_split(bag_of_words, wikipedia_y,random_state=0)
```

In [8]:
```
SVCfit  = modelSVC.fit(x_train,y_train)
NBfit   = modelNB.fit(x_train,y_train)

print (SVCfit)
print (NBfit)
```

```
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)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

classify

Text Classification on Scikit-learn

In [9]:
```
pred_SVC =SVCfit.predict(x_test)
pred_NB  =NBfit.predict(x_test)
```

In [14]:
```
accuracySVC=accuracy_score(pred_SVC,y_test)
accuracyNB=accuracy_score(pred_NB,y_test)
print("Accuracy of LinearSVC : %",accuracySVC*100)
print("Accuracy of MultinomialNB : %",accuracyNB*100)
```

```
Accuracy of LinearSVC : % 90.66666666666666
Accuracy of MultinomialNB : % 92.0
```

Classification Accuracy with k-fold Cross Validation (k=10)

In [4]:
```
wikipedia_y=wikipedia["Category"]
wikipedia_x=wikipedia["Text"]
```

In [5]:
```
cv=CountVectorizer()
bag_of_words =cv.fit_transform(wikipedia_x)
```

In [6]:
```
x_train,x_test,y_train,y_test =train_test_split(bag_of_words, wikipedia_y,random_state=0)
```

In [7]:
```
accuracySVC = cross_val_score(modelSVC, x_train, y_train , scoring='accuracy', cv = 10)
accuracyNB = cross_val_score(modelNB,   x_train, y_train , scoring='accuracy', cv = 10)
```

In [9]:
```
print("Accuracy of Model LinearSVC with Cross Validation=10 is: %",accuracySVC.mean() * 100)
print("Accuracy of ModelMultinomialNB with Cross Validation=10 is: %",accuracyNB.mean() * 100)
```

```
Accuracy of Model LinearSVC with Cross Validation=10 is: % 92.4703557312253
Accuracy of ModelMultinomialNB with Cross Validation=10 is: % 96.0079051383399
```

← accuracy with cross validation

TF-IDF based Text Classification

```
In [4]: wikipedia_y=wikipedia["Category"]
        wikipedia_x=wikipedia["Text"]

In [5]: cv=CountVectorizer()
        bag_of_words =cv.fit_transform(wikipedia_x)

In [6]: x_train,x_test,y_train,y_test =train_test_split(bag_of_words, wikipedia_y,random_state=0)

In [7]: from sklearn.feature_extraction.text import TfidfTransformer

In [8]: TFIDF= TfidfTransformer()
        x_train_tfidf = TFIDF.fit_transform(x_train)

In [9]: accuracySVC = cross_val_score(modelSVC, x_train_tfidf, y_train , scoring='accuracy', cv = 10)
        accuracyNB = cross_val_score(modelNB,   x_train_tfidf, y_train , scoring='accuracy', cv = 10)

In [10]: print("Accuracy of Model LinearSVC with Cross Validation=10 is: %",accuracySVC.mean() * 100)
         print("Accuracy of ModelMultinomialNB with Cross Validation=10 is: %",accuracyNB.mean() * 100)

Accuracy of Model LinearSVC with Cross Validation=10 is: % 95.57312252964427
Accuracy of ModelMultinomialNB with Cross Validation=10 is: % 95.55335968379445
```

TF-IDF

TF-IDF ACCURACY

Classification on Iris dataset on Scikit-learn

- Classify the Iris dataset in Scikit using the k-Nearest Neighbour algorithm.

- Experiment with different values for k. Which setting gives the best accuracy?

- Test and compare the results when using a Decision Tree classifier instead. Which gives the best accuracy.

Classification on Iris dataset on Scikit-learn

In [128]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.datasets import load_iris
```

In [129]:

```
iris=load_iris()
```

In [130]:

```
print(iris)
```

```
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
```

Classification on Iris dataset on Scikit-learn

In [131]:

```
print(iris.data[0:5])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

Classification on Iris dataset on Scikit-learn

For k=3

```
In [132]:  print(iris.target)

[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 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]

In [133]:  x1,x2,y1,y2 =train_test_split(iris.data, iris.target, test_size =0.3)

In [164]:  KN=KNeighborsClassifier(n_neighbors=3)

In [165]:  print(KN)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')

In [166]:  KN.fit(x1,y1)

Out[166]:  KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')

In [167]:  y_prediction=KN.predict(x2)

In [168]:  print("ACCURACY=",metrics.accuracy_score(y2,y_prediction))

ACCURACY= 0.9777777777777777
```

Classification on Iris dataset on Scikit-learn

For k=11

In [184]:
```
print(iris.target)
```
```
[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 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

In [185]:
```
x1,x2,y1,y2 =train_test_split(iris.data, iris.target, test_size =0.3)
```

In [186]:
```
KN=KNeighborsClassifier(n_neighbors=11)
```

In [187]:
```
print(KN)
```
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=11, p=2,
                     weights='uniform')
```

In [188]:
```
KN.fit(x1,y1)
```

Out[188]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=11, p=2,
                     weights='uniform')
```

In [189]:
```
y_prediction=KN.predict(x2)
```

In [190]:
```
print("ACCURACY=",metrics.accuracy_score(y2,y_prediction))
```
```
ACCURACY= 0.9555555555555556
```

Regression on Scikit-learn

- House Sales in King Country
- https://www.kaggle.com/harlfoxem/housesalesprediction
- Utilize linear regression models available in Scikit-learn

Regression on Scikit-learn

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression , Lasso , Ridge
from sklearn.svm import LinearSVR
from sklearn.tree import DecisionTreeRegressor
```

In [2]:

```
house=pd.read_csv("C:\\Users  \\Desktop\\house_data.csv")
```

Regression on Scikit-learn

In [3]:
```
house.describe()
```

Out[3]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	216
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	

Regression on Scikit-learn

In [15]:

```
models = [
    ('House Price Accuracy with Linear Regression', LinearRegression()),
    ('House Price Accuracy with Lasso', Lasso()),
    ('House Price Accuracy with Ridge', Ridge()),
    ('House Price Accuracy with LinearSVR', LinearSVR()),
    ('House Price Accuracy with Decision Tree Regressor', DecisionTreeRegressor()),
]
```

regression models

Regression on Scikit-learn

In [22]:

```
X = house.iloc[:,[3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]].values
y = house.iloc[:,2].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
for name, model in models:
    clf = model
    clf.fit(X_train, y_train)
    accuracy = clf.score(X_test, y_test)
    print(name,"= %", accuracy*100)
```

```
House Price Accuracy with Linear Regression = % 69.89355410059093
House Price Accuracy with Lasso = % 69.89378872470675
House Price Accuracy with Ridge = % 69.90063019439276

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\coordinate_descent.py:475: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 324412975407149.6, tolerance: 236908691299.8883
  positive)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:929: Convergenc
se the number of iterations.
  "the number of iterations.", ConvergenceWarning)

House Price Accuracy with LinearSVR = % 50.400638597908696
House Price Accuracy with Decision Tree Regressor = % 74.00178091998895
```

regression models predicts