Features Selection

Goal

The goal of feature selection in machine learning is to find the best set of features that allows one to build useful models of studied phenomena.

1 Feature Selection with Univariate Statistical Tests

In [1]:
```
import pandas as pd
import numpy as np
from numpy import set_printoptions
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.feature_extraction.text import CountVectorizer
df = pd.read_csv('nlp_emails.csv')

X = np.array(df['text'])
Y = np.array(df['spam'])

cv = CountVectorizer()

X = cv.fit_transform(X)
```

In [3]:
```
# feature extraction
test = SelectKBest(score_func=f_classif, k=5)
fit = test.fit(X, Y)
```

In [4]:
```
# summarize scores
set_printoptions(precision=5)
print(fit.scores_)
```

```
[ 4.93319 12.72295 17.99644 ...  0.3493   0.3493   2.86635]
```

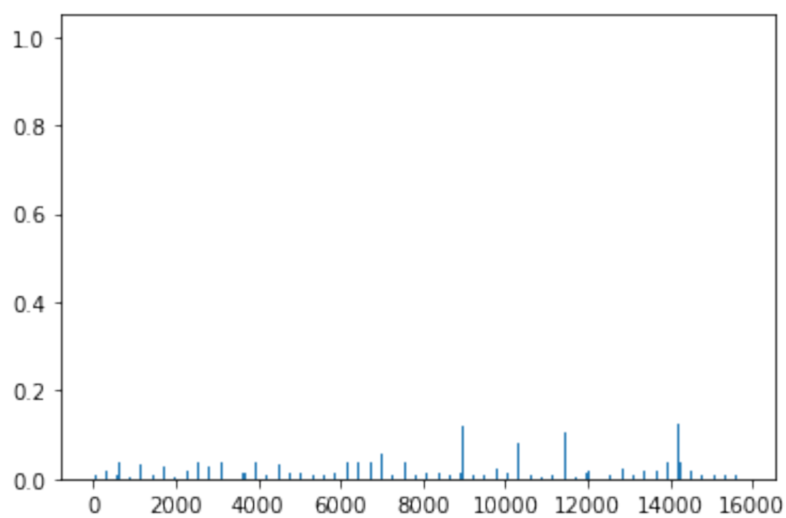
In [5]:
```
features = fit.transform(X)
# summarize selected features
print(features[0:5,:])
```

```
  (0, 4)	1
  (0, 2)	13
  (0, 3)	1
  (0, 0)	1
  (2, 4)	1
  (3, 4)	5
  (3, 2)	18
  (3, 3)	3
  (3, 0)	3
  (4, 4)	3
  (4, 3)	1
```

In [6]:
```
# Univariate feature selection with F-test for feature scoring
# We use the default selection function to select the four
# most significant features
X_indices = np.arange(X.shape[-1])

scores = -np.log10(test.pvalues_)
scores /= scores.max()
plt.bar(X_indices - .45, scores, width=.2,
        label=r'Univariate score ($-Log(p_{value})$)')
```

Out[6]:
```
<BarContainer object of 15771 artists>
```



In [7]:
```
from sklearn.feature_selection import SelectKBest, chi2
chi2_test = SelectKBest(chi2, k=5)
X_new = chi2_test.fit_transform(X, Y)
```

In [8]:
```
# summarize selected features
print(X_new[0:5,:])
```

```
  (0, 4)	1
  (0, 2)	13
  (2, 4)	1
  (3, 4)	5
  (3, 2)	18
  (3, 3)	1
  (3, 1)	1
  (4, 4)	3
```

In [9]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import SelectPercentile, chi2
X_cat = X.astype(int)
chi2_features = SelectKBest(chi2, k = 4)
X_kbest_features = chi2_features.fit_transform(X_cat,Y)

print("original feature number:", X_cat.shape[1])
print('Reduced feature number:', X_kbest_features.shape[1])
```

```
original feature number: 15771
Reduced feature number: 4
```

2 Recursive Feature Elimination

The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain.

It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

The example below uses RFE with the logistic regression algorithm to select the top 4 features automaticaly. The choice of algorithm does not matter too much as long as it is skillful and consistent.

In [10]:
```
# Feature Extraction with RFE

from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
#from sklearn.naive_bayes import GaussianNB
```

In [11]:
```
# feature extraction
df = pd.read_csv('nlp_emails.csv')

X = np.array(df['text'])
Y = np.array(df['spam'])

cv = CountVectorizer(max_features = 1500)

X = cv.fit_transform(X)

#model = GaussianNB()

model = LogisticRegression()
rfe = RFE(model, n_features_to_select=4)
fit = rfe.fit(X, Y)
print("Num Features: %d" % fit.n_features_)
```

```
Num Features: 4
```

In [12]:
```
print("Selected Features: %s" % fit.support_)
```

```
Selected Features: [False False False ... False False False]
```

In [13]:
```
print("Feature Ranking: %s" % fit.ranking_)
```

```
Feature Ranking: [  37  174 1409 ...   59  621  135]
```

In [14]:
```
from sklearn.feature_selection import VarianceThreshold

V_threshold = VarianceThreshold(threshold = 0)
V_threshold.fit(X)# fit finds the feature with zeron variance
V_threshold.get_support()
```

Out[14]:
```
array([ True,  True,  True, ...,  True,  True,  True])
```

In [15]:
```
print(__doc__)

import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_selection import RFECV
from sklearn.datasets import make_classification

# Build a classification task using 3 informative features
X, Y = make_classification(n_samples=1000, n_features=25, n_informativ
                           n_redundant=2, n_repeated=0, n_classes=8,
                           n_clusters_per_class=1, random_state=0)
```

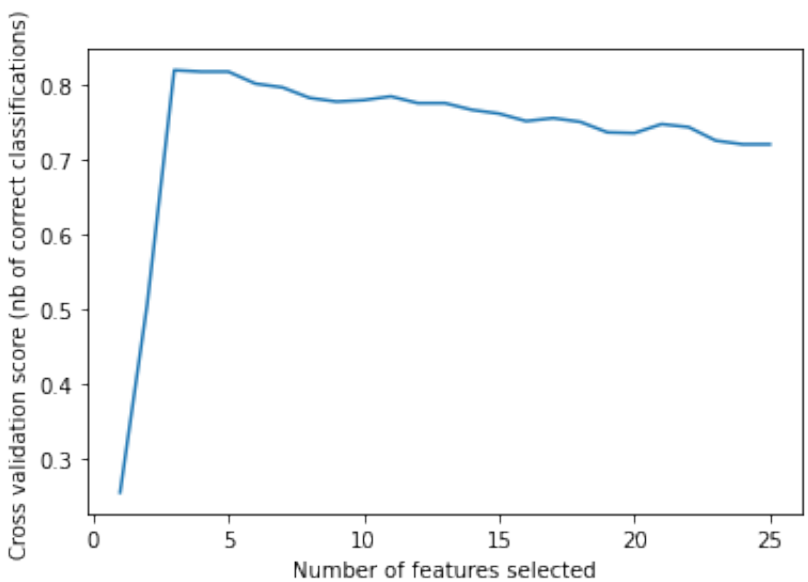
```python
# Create the RFE object and compute a cross-validated score.
svc = SVC(kernel="linear")
# The "accuracy" scoring is proportional to the number of correct
# classifications

min_features_to_select = 1  # Minimum number of features to consider
rfecv = RFECV(estimator=svc, step=1, cv=StratifiedKFold(2),
              scoring='accuracy',
              min_features_to_select=min_features_to_select)
rfecv.fit(X, Y)

print("Optimal number of features : %d" % rfecv.n_features_)

# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(min_features_to_select,
               len(rfecv.grid_scores_) + min_features_to_select),
         rfecv.grid_scores_)
plt.show()
```

```
Automatically created module for IPython interactive environment
Optimal number of features : 3
```



3 Principal Component Analysis

Principal Component Analysis (or PCA) uses linear algebra to transform the dataset into a compressed form.

Generally this is called a data reduction technique. A property of PCA is that you can choose the number of dimensions or principal component in the transformed result.

In the example below, we use PCA and select 3 principal components.

Learn more about the PCA class in scikit-learn by reviewing the PCA API. Dive deeper into the math behind PCA on the Principal Component Analysis Wikipedia article.

In [16]:

```
# Feature Extraction with PCA

from sklearn.decomposition import PCA

# feature extraction
pca = PCA(n_components=2)
fit = pca.fit(X)
# summarize components
print("Explained Variance: %s" % fit.explained_variance_ratio_)
```

```
Explained Variance: [0.18979 0.06567]
```

In [17]:

```
print(fit.components_)
```

```
[[-0.00568  0.43175  0.00706 -0.01666 -0.00238  0.01096  0.02261  0.4
3865
  -0.70516  0.00938 -0.02536  0.01259  0.02457  0.00459  0.00716 -0.0
0133
  -0.00388 -0.00338 -0.05176 -0.00688  0.00515  0.0173   0.34278  0.0
2319
   0.01565]
 [-0.01859 -0.6104  -0.02646 -0.02399 -0.06365  0.03408  0.00443  0.3
0685
  -0.00596  0.00927  0.05788 -0.0587   0.02691 -0.04708  0.06861  0.0
1865
   0.01571 -0.05832 -0.64794  0.0784   0.07512  0.05549  0.26632 -0.0
441
   0.01659]]
```

4 Feature Importance

Bagged decision trees like Random Forest and Extra Trees can be used to estimate the importance of features.

In the example below we construct a ExtraTreesClassifier classifier for the Pima Indians onset of diabetes dataset. You can learn more about the ExtraTreesClassifier class in the scikit-learn API.

In [18]:
```
# Feature Importance with Extra Trees Classifier

from sklearn.ensemble import ExtraTreesClassifier

# feature extraction
model = ExtraTreesClassifier(n_estimators=10)
model.fit(X, Y)
print(model.feature_importances_)
```

```
[0.02285 0.12574 0.01896 0.01951 0.02148 0.02369 0.02214 0.1007  0.10
054
 0.02265 0.02252 0.01776 0.02118 0.02007 0.01955 0.01841 0.01866 0.02
242
 0.1562  0.01805 0.02045 0.02013 0.10471 0.01939 0.02222]
```

Spam Emails Detection Machine Learnig Lifecycle Summary

- Data Preprocessing
- Feature Engineering
- Models Training
- Models Evaluation
- Models Selection and Features Selection
- Model Deployment

In [19]:
```
import warnings
warnings.filterwarnings("ignore")

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [20]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report
```

Import data and text preprocessing

Scape the text cleaning: During the practice of this project, I found that without cleaning the text, use 'latin-1' encoding. The TfidVectorizer which has "stopword" itself returns the best result. Probably it makes sence because many spam emails has a lot of strange symbols, meaningless words.By keeping the origianl text, it will help the machine filter the spam emails better.

TfidVectorizer combines the CountVectorizer and TfidTransformer steps into one using TfidVectorizer.

LinearSVC handles sparse input better, and scales well to large numbers of samples.

Using **train_test_split** method

In [21]:
```
df = pd.read_csv('Emails.csv', encoding="latin-1")# encoding='ISO-8859

X = df['text']
y = df['spam']

tv = TfidfVectorizer()
X = tv.fit_transform(X) # Fit the Data


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

clf = LinearSVC()
clf.fit(X_train,y_train)
clf.score(X_test,y_test)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

Out[21]:
```
LinearSVC()
```

Out[21]:
```
0.9894235854045479
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1398
           1       0.99      0.97      0.98       493

    accuracy                           0.99      1891
   macro avg       0.99      0.98      0.99      1891
weighted avg       0.99      0.99      0.99      1891
```

Spam emails are the minority cases so that using **Stratified sampling** will be better

In [22]:
```
from sklearn.model_selection import train_test_split, StratifiedShuffl


test_size = np.arange(0.05, 0.55, 0.05)


for sz in test_size:
    #stratified sampling
    sss = StratifiedShuffleSplit(n_splits=1, test_size=sz, random_stat

    #train-test split
    for trn_idx, tst_idx in sss.split(X, y):
        x_trn, y_trn, x_tst, y_tst = X[trn_idx], y[trn_idx], X[tst_idx
            tst_idx]
```

```
    #model fitting
    lsvc_clf = LinearSVC()
    lsvc_clf.fit(x_trn, y_trn)

    lsvc_clf.score(x_tst,y_tst)
    y_pred = lsvc_clf.predict(x_tst)
print(classification_report(y_tst, y_pred))
```

```
Out[22]: LinearSVC()

Out[22]: 0.9965156794425087

Out[22]: LinearSVC()

Out[22]: 0.9982547993019197

Out[22]: LinearSVC()

Out[22]: 0.9988372093023256

Out[22]: LinearSVC()

Out[22]: 0.9973821989528796

Out[22]: LinearSVC()

Out[22]: 0.994413407821229

Out[22]: LinearSVC()

Out[22]: 0.9941826643397323

Out[22]: LinearSVC()

Out[22]: 0.9920199501246882

Out[22]: LinearSVC()

Out[22]: 0.993891797556719

Out[22]: LinearSVC()

Out[22]: 0.9922420480993018

Out[22]: LinearSVC()

Out[22]: 0.9912709497206704
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2180
           1       1.00      0.96      0.98       684

    accuracy                           0.99      2864
```

```
   macro avg       0.99      0.98      0.99      2864
weighted avg       0.99      0.99      0.99      2864
```

In [23]:
```
df['prediction']= lsvc_clf.predict(X)
```

Check what emails are spam but detected as non_spam

In [24]:
```
sneaky_spam = df[(df['prediction']==0) & (df['spam']==1)]['text']
for email in sneaky_spam:
    print(email)
```

```
Subject: use this handy interest calculator to get current rate inf
ormation . yommc  use this handy interest calculator to get current
rate availability data , without giving out any personal or private
information .  this was sent to you by an mediagroup for smartmortg
ageusa . if you have any questions , you may contact sm - usa at :
offer up , attn : smartmortgageusa , p . o . box 78361 , san franci
sco , ca 94107 - 8361 . if you wish to exclude yourself from future
sm - usa items please use this to go to the website and then use th
e choice at the bottom of the page .  wwcidawgmcln
Subject: cool page  please active this link www . informatii . as .
ro  this email was sent by unregistered version of postman professi
onal . please visit : www . email - business . com
Subject: reduction in high blood pressure  age should be nothing mo
re than a number  it ' s okay to want to hold on to your young body
as long as you can  view more about a new  lifespan enhancement pre
ss here  with increasing longevity for an increasing segment of the
population , this is the frontier for the new millennium  - dr davi
d howard  medical journal news  sorry not for me and the address is
above  this was good reasoning , but the rash youth had no idea he
```

Check what emails are non_spam but detected as spam emails

In [25]:
```
not_actually_spam = df[(df['prediction']==1) & (df['spam']==0)]['text'
for email in not_actually_spam:
    print(email)
```

```
Subject: investment  to all whom it may concern :  i keep receiving s
uch unsolicited messages  ( please , see below ) that represent an ob
vious scam .  is there any way to block out the messages from this so
urce ?  i assume other employees of enron are also being hit .  vince
kaminski  - - - - - - - - - - - - - - - - - - - - - - forwarded by vi
nce j kaminski / hou / ect on 01 / 02 / 2001  08 : 19 am - - - - - -
- - - - - - - - - - - - - - - - - - - - -  " dagogo kalu " on 12 / 30
/ 2000 02 : 15 : 58 am  to : vince . j . kaminski @ enron . com  cc :
subject : investment  from : col . dagogo kalu  tel / fax 44845334278
3  attn : president / ceo  request for an assistance  dear sir ,  wit
```

```
h due respect and knowledge of your good reputation , i decided to co
ntact  you for this matter that need urgent attention . i am col . da
gogo kalu . i  was the commandant of the west african peace keeping f
orce & monitoring  group ( ecomog ) until i sustained a very serious 
injury that put me off the  military camp for about 3 months now . ho
nestly , i have full trust for your  honesty hence i write this lette
r to you .  during our recent mission to sierra leone , a diamond ric
h west african  country to cushion the war between the rebels and the
ruling government , we  ran into 2 boxes ( consignment ) right inside
a thick jungle where we believed  was a hide out of the rebels . as w
e opened the boxes , one that is smaller  contains numerous sizes of 
raw diamond . the bigger box contains about us  $ 15 . 2 million , wh
ich we believed to be the total amount of diamond sold at  that perio
d of time by the rebels before we invaded the place .  myself and my 
two other colleagues took the boxes away . i took the bigger  box con
taining the us $ 15 . 2 million to cotonou benin republic which is th
e  nearby country and deposited it with one security company for safe
keeping  to enable me think wisely on what to do with the money . the
smaller box  containing the raw diamond i gave it to my other 2 colle
agues , which they  accepted in good faith .  a month after this , th
e rebels lunched a counter - attack on us where i  sustained a very s
erious gun shut wound on my right leg . a lot of our boys  died but f
ew survived . at this moment , i am receiving medical treatment  here
in london . this is why i need your help urgently .  i need a foreign
company i will present as the beneficiary of the huge  amount and als
o pay in the money into their account . your bank account must  be a 
good one where we shall not pay much as tax . you shall also serve as
the general overseer and guardian of this fund and all the investment
of  this money will be under your care until i recover fully .  i wil
l give you all the details of the transaction immediately i get your 
response . you will also get a suitable percentage ( % ) as your shar
e . all  the documents of the deposit of the money are intact with my
wife . i have  already finalized this arrangement with the security c
ompany so there is no  problem at all .  please you can reach me thro
ugh my direct tel / fax no : 448453342783 where i  am receiving treat
ment . in your reply , state clearly your direct telephone  and fax n
umbers for easy communication and more confidentiality .  further det
ails will be given to you once i hear from you .  i await your urgent
response soonest .  best regards ,  col . dagogo kalu  get your free 
download of msn explorer at http : / / explorer . msn . com
```

Using **PipeLine** to creat a spam email detection model to do new emails detection

Using **StratifiedShuffles** to split the train/test dataset

In [33]:
```
df = pd.read_csv('Emails.csv', encoding="latin-1")# encoding='ISO-8859

X = df['text']
y = df['spam']

test_size = np.arange(0.05, 0.55, 0.05)


for sz in test_size:
    #stratified sampling
    sss = StratifiedShuffleSplit(n_splits=1, test_size=sz, random_stat

    #train-test split
    for trn_idx, tst_idx in sss.split(X, y):
        x_trn, y_trn, x_tst, y_tst = X[trn_idx], y[trn_idx], X[tst_idx
            tst_idx]


from sklearn.pipeline import Pipeline

text_clf = Pipeline([('tfidf', TfidfVectorizer()),('clf',LinearSVC())]

text_clf.fit(x_trn,y_trn)

predictions = text_clf.predict(x_tst)

print(classification_report(y_tst,predictions))
```

Out[33]:
```
Pipeline(steps=[('tfidf', TfidfVectorizer()), ('clf', LinearSVC())])

              precision    recall  f1-score   support

           0       0.99      1.00      0.99      2180
           1       1.00      0.97      0.98       684

    accuracy                           0.99      2864
   macro avg       0.99      0.98      0.99      2864
weighted avg       0.99      0.99      0.99      2864
```

In [28]:
```
text = "Congratulation! You win a prize! Please provide your bank accc


text_clf.predict([text])
```

Out[28]:
```
array([1])
```

The email is predicted as a spam email.

In [29]:
```
text = "Hello, I am taking a Machine Learning course in York Universit


text_clf.predict([text])
```

Out[29]: array([0])

Save text_clf model for future model deployment

In [31]:
```

import joblib
joblib.dump(text_clf, 'spam-detection-model,pkl')
```

Out[31]: ['spam-detection-model,pkl']

In [32]:
```
spam_LSVC_model = open('spam-detection-model,pkl','rb')
spam_clf = joblib.load(spam_LSVC_model)
```

Conclusion

I followed the steps of nature language process NLP machine learning lifecycle. I found that the test prediction result better without cleaning the emails text, just use 'latin-1' encoding. In real life, saving the time of processing the spam detection is very important. So that the text for final model are not cleaned. The methods of TfidfVectorizer for vectorization plays better result. I made around 10 models and found that LinearSVC turns the highest scores. The precision is one of the important metric to determine if the model is very good for spam emails detection, because we should avoid non_spam emails filtered to the spam box. In this project: spam emails are the minority cases so that using Stratified sampling will be better.