In [143]:

```
import numpy as np
import pandas as pd
```

In [144]:

```
df = pd.read_csv('C:/Users/jry5/OneDrive/Desktop/make_up_data.csv')
```

In [145]:

```
df.head()
```

Out[145]:

	problem_id	problem_detail	concepts
0	1	True/False, we can always derive the estimates...	calling APIs, joins
1	2	If I multiply every Y value by a constant a , ...	joins, regression
2	3	If I add a constant b to every Y value, what h...	joins, estimation
3	4	If I multiply every X value by a constant a , ...	interpretation, regression
4	5	If I add a constant b to every X value, what h...	interpretation, estimation

In [146]:

```
import re
def clean_text(text):
    # remove backslash-apostrophe
    text = re.sub("\'", "", text)
    # remove everything except alphabets
    text = re.sub("[^a-zA-Z]"," ",text)
    # remove whitespaces
    text = ' '.join(text.split())
    # convert text to lowercase
    text = text.lower()

    return text
```

In [147]:

```
df['clean_detail'] = df['problem_detail'].apply(lambda x: clean_text(x))
```

In [160]:

```
df.clean_detail[0]
```

Out[160]:

```
'true false we can always derive the estimates for the regression line slope and i
ntercept with only summary statistics and not the granular data points'
```

In [161]:

```
from sklearn.preprocessing import MultiLabelBinarizer
ohe = MultiLabelBinarizer()
encoding_detail = ohe.fit_transform(df['concepts'].str.split(', '))
encoding_class = ohe.classes_
print(encoding_detail)
print(encoding_class)
```

```
[[1 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 1 0 0 0]
 [0 0 0 0 1 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 1 0 0 0]
 [0 0 0 0 1 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 1 0 0 0]
 [0 0 0 0 0 0 1 1 0 0 0 0 0]
 [0 0 0 0 1 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 1]
 [0 0 0 0 0 0 0 0 0 1 0 0 1]
 [0 0 0 0 0 0 0 0 0 1 0 0 1]
 [0 0 0 0 1 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 1 0 1 0]
 [0 0 0 0 0 0 0 1 1 1 0 0 0]
 [0 1 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 1]
 [0 0 0 0 0 0 0 1 0 0 0 1 0]
 [1 0 0 0 0 0 0 1 0 0 0 1 0]
 [1 0 0 0 0 0 0 1 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 1 1 0 0 0]
 [0 0 0 1 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 1 0]
 [0 0 1 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 1]
 [0 0 1 0 0 0 0 0 0 1 0 0 0]
 [0 0 1 0 0 0 0 0 0 1 0 0 0]
 [0 0 1 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 1 0 0 0]]
['calling APIs' 'data generation' 'diagnostic' 'distribution' 'estimation'
 'expectation' 'interpretation' 'joins' 'probability' 'regression'
 'simulation' 'thinking about wrong models' 'validation']
```

In [162]:

```
df_ohe = pd.DataFrame(encoding_detail,
                      columns=encoding_class)
```

In [163]:

```
df_convert = df.join(df_ohe)
df_convert.head()
```

Out[163]:

	problem_id	problem_detail	concepts	clean_detail	predict_concepts	calling APIs	da generatic
0	1	True/False, we can always derive the estimates...	calling APIs, joins	true false we can always derive the estimates ...	calling APIs, joins	1	
1	2	If I multiply every Y value by a constant a , ...	joins, regression	if i multiply every y value by a constant a wh...	joins, regression	0	
2	3	If I add a constant b to every Y value, what h...	joins, estimation	if i add a constant b to every y value what ha...	estimation, joins	0	
3	4	If I multiply every X value by a constant a , ...	interpretation, regression	if i multiply every x value by a constant a wh...	joins, regression	0	
4	5	If I add a constant b to every X value, what h...	interpretation, estimation	if i add a constant b to every x value what ha...	estimation, joins	0	

In [164]:

```
# Transform Descriptions using TfIdf
# Vectorize from strings to fixed vectors of floats.

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(lowercase=True,
                        ngram_range=(1,2),
                        stop_words='english')
X_tfidf = tfidf.fit_transform(df_convert.loc[:, 'clean_detail'])
y_train = encoding_detail
X_tfidf.shape
```

Out[164]:

```
(30, 379)
```

In [165]:

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC

clf = OneVsRestClassifier(LinearSVC())
model = clf.fit(X_tfidf, y_train)
output = model.predict(X_tfidf)
output
```

Out[165]:

```
array([[1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0],
       [0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0]])
```

In [166]:

```
# training score
training_score = model.score(X_tfidf, y_train)
training_score
```

Out[166]:

```
0.9333333333333333
```

In [167]:

```
df_output = pd.DataFrame(output,
                         columns=encoding_class)

# return as list
# df['predict_concepts'] = df_output.apply(lambda x: x.index[x.astype(bool)].tolist(), 1)

#return as string
s = np.where(df_output, ['{}, '.format(x) for x in df_output.columns], '')
df['predict_concepts'] = pd.Series([''.join(x).strip(', ') for x in s], index=df.index)
```

In [168]:

```
df.head(10)
```

Out[168]:

	problem_id	problem_detail	concepts	clean_detail	predict_concepts
0	1	True/False, we can always derive the estimates...	calling APIs, joins	true false we can always derive the estimates ...	calling APIs, joins
1	2	If I multiply every Y value by a constant a , ...	joins, regression	if i multiply every y value by a constant a wh...	joins, regression
2	3	If I add a constant b to every Y value, what h...	joins, estimation	if i add a constant b to every y value what ha...	estimation, joins
3	4	If I multiply every X value by a constant a , ...	interpretation, regression	if i multiply every x value by a constant a wh...	joins, regression
4	5	If I add a constant b to every X value, what h...	interpretation, estimation	if i add a constant b to every x value what ha...	estimation, joins
5	6	Show that the sum of the residuals from SLR is...	joins, regression	show that the sum of the residuals from slr is...	joins, regression
6	7	Choose 10 adjectives randomly from the list ab...	interpretation, joins	choose adjectives randomly from the list above...	interpretation, joins
7	8	Now fit the linear regression by fitting point...	estimation, joins	now fit the linear regression by fitting point...	estimation, joins
8	9	Your team about to launch some marketing descr...	validation, regression	your team about to launch some marketing descr...	regression, validation
9	10	Now your boss wants to communicate out your pr...	validation, regression	now your boss wants to communicate out your pr...	regression, validation

In [177]:

```
# More to do:
# training/test
# bootstrap?
```